Benutzerhandbuch für .CEL-Scripts

für Version 1.3.1

Überarbeitet von Selden Ball, Jr. und anderen Forum-Mitgliedern.
Version 1.0d
Letzte Aktualisierung: 04.01.2004

Ins Deutsche übersetzt und überarbeitet © von Ulrich „Adirondack“ Dickmann, im April 2004
http://www.celestia.de.vu/

Inhaltsverzeichnis

Vorwort

Dieses deutsche Handbuch zum Erstellen von eigenen Scrips zur Verwendung in Celestia wurde auf Grundlage von Don Goyette's „Scripting Guide" (Stand: 04.01.2004) und mit seiner Zustimmung erstellt.
Sein „Scripting Guide" kann im Original (englischsprachig) über die Celestia-Website unter http://www.shatters.net/documentation.html heruntergeladen werden.
Dort bzw. auf seiner Website steht es in verschiedenen Dateiformaten (HTML, PDF und MS Word) jeweils in der aktuellen Fassung zur Verfügung.

Ich habe auf diverse Dateiformate für das deutsche Handbuch verzichtet und stelle es ausschließlich als PDF bereit. Ich habe auch nicht immer eine 1:1 Übersetzung vorgenommen, sondern mich hier auf das Wesentliche (nämlich auf das Scripting) beschränkt. Gelegentlich finden Sie auch meine persönlichen Anmerkungen, die mit „Anm. d. Übers.:" beginnen und **farbig** dargestellt sind.

Mein Dank gilt Don Goyette, ohne dessen Vorarbeit in Form der englischsprachigen Fassung dieses deutsche Handbuch nicht möglich geworden wäre.

Weitere (meist englischsprachige) Informationen zum Scripting finden Sie auch im Celestia-Forum unter http://www.shatters.net/forum/viewforum.php?f=9.

Obwohl Don und alle Editoren alle Informationen und Code-Beispiele in diesem Handbuch gewissenhaft zusammengetragen haben, kann nicht ausgeschlossen werden, dass Tipp- oder andere Fehler in den Scriptbeispielen möglich sein können.

Bitte beachten Sie, dass ich keine persönlichen Hilfestellungen zum Scripting geben kann. Bei Fragen wenden Sie sich daher bitte an das Celestia-Scripting-Forum (Direktzugriff über http://www.shatters.net/forum/viewforum.php?f=9, in dem Ihnen meist sehr schnell geholfen wird.

Einführung

Wenn Sie Neuling bei Celestia sind, sollten Sie sich zunächst das Benutzerhandbuch zu Celestia 1.3.1 durchlesen. Es ist ebenfalls auf der Celestia-Website unter der Internet-Adresse http://www.shatters.net/celestia/documentation.html in verschiedenen Sprachen erhältlich:

English (MS Word, HTML und PDF)
Deutsch (MS Word und PDF)
Französisch (MS Word)
Japanisch (MS Word und TXT)

In diesem Benutzerhandbuch zu Celestia 1.3.1 erfahren Sie die grundsätzliche Handhabung von Celestia. Wenn Sie sich mit der Handhabung des Programms schließlich vertraut gemacht haben, möchten Sie vielleicht Ihre Reisen durch das Celestia-Universum auch anderen zugänglich machen. Um dies zu bewerkstelligen gibt es in Celestia das sogenannte .CEL-Scripting. Und genau darum, geht es in diesem Ihnen nun vorliegenden Handbuch.

Wenn Sie finden, dass das .CEL-Scripting viel zu einfach ist, möchten Sie vielleicht auch noch in die Lua-Programmiersprache einsteigen, mit der man seit der Celestia Version 1.3.1 ebenfalls Scripte einbinden kann.
Dieses Scripting-Handbuch befasst sich jedoch nicht mit diesen Scripts.
Wenn Sie mehr zu Lua erfahren möchten, finden Sie unter http://www.lua.org/ weitere Informationen. Einige Beispiele zu Lua-Scripts finden Sie hier:
http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/celestia/celestia/scripts/

Außerdem gibt es von Harald Schmidt ein PDF-Dokument mit dem Namen „*Summary of Lua-support in Celestia*" auf dessen Celestia-Site http://www.h-schmidt.net/celestia/

Anm. d. Übers.: Diese Lua-Scripts (CELX-Scripts) werden in Zukunft wahrscheinlich die CEL-Scripts ablösen, denn die CELX-Scripte sind flexibler und lassen interaktive Eingriffe des Benutzers bei der Ausführung der Scripte zu. Ich gehe davon aus, dass spätestens nach der Veröffentlichung von Celestia 1.3.2 derartige Scripte vermehrt zum Einsatz kommen werden. Nach Mitteilung der Celestia-Entwickler werden aber auch zukünftig CEL-Scripte von Celestia unterstützt. Das Wissen, das Sie sich mit diesem Handbuch aneignen ist also auch in zukünftigen Celestia-Versionen noch brauchbar.

Symbole im Handbuch



Im Handbuch verwendete Bezeichnungen

Argument

Ein Argument bezieht sich entweder auf eine Variable oder auf ein Parameter. Dabei ist das Argument auch für Menschen lesbar und verständlich. Der Wert, den Sie dem Argument zuweisen wird im Speicher des Computers abgelegt. So hat z.B. das Kommando `wait` ein Argument, dass sich `duration` nennt. Andere Kommandos, wie beispielsweise `print`, können mehr als nur ein Argument haben.

Datentypen

Ein *Datentyp* beschreibt den Typ oder die Art der Daten, die von einem Argument benutzt werden. Die in Celestia benutzen Datentypen sind **<String>**, **<Wert>** und **<Vector>**:

<String> Dieser Datentyp besteht aus einer Zeichenfolge, die in Anführungsstrichen (") eingeschlossen ist.

Beispiel:

```
select { object "Mars" }
```

Manche Kommandos erfordern einen <String>-Wert, der in einer internen Liste abgespeichert ist! Andere Werte werden daher in einem solchen Fall zurückgewiesen. Beispielsweise benutzt das Kommando `set` ein Argument mit dem Namen `name`, das ein <String>-Wert ist und einen der nachfolgenden Werte haben **muss**:

```
MinOrbitSize
AmbientLightLevel
FOV
StarDistanceLimit
```

Wenn Sie also einen Text zum Argument `name` angeben, der nicht den o.g. Werten entspricht (also etwa “Hallo”), erhalten Sie einen Scriptfehler und das Script funktioniert nicht.

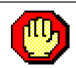
<Wert> Dieser Datentyp besteht nur aus den Ziffern 0 – 9. Er kann (muss aber nicht) auch folgende argumentabhängige Daten enthalten:

* Dezimalpunkt (wie etwa 5.5)
* Vorzeichen ("-" für negative Werte, wie etwa -98.53)
* Expotentzeichen (wie etwa 31.0e+15)

Wenn Sie einen <Wert> zuweisen, benutzen Sie bitte keine Anführungszeichen (wie es bei den <String>-Werten der Fall sein muss)!

Beispiel:

```
wait { duration 5.5 }
```



** Benutzen Sie **keine negativen** Werte für das `time` Argument oder das `duration` Argument.*

<Vector> Dieser Datentyp besteht aus einer Ansammlung (oder Gruppe) von drei individuellen Werten, die in eckigen Klammern eingeschlossen sind. Jeder Wert wird dabei mit einem Leerzeichen getrennt.

Der <Vector>-Datentyp in Celestia wird meist benutzt, um dreidimensionale Koordinaten mit drei Werten auf den X, Y und Z-Achsen zu setzen. Je nach Argument repräsentieren die X, Y und Z Werte verschiedene Dinge.

Beispiel:

Das `setposition` Kommando erfordert die beiden <Vector>-Werte `base` und `offset`. Die beiden unten aufgeführten Werte positionieren die Ansicht außerhalb der Milchstrasse:

```
setposition {
base [-64132.43 47355.11 196091.57 ]
offset [ 0 0 -1.52e-005 ] }
```

In einigen Kommandos spezifiziert der <Vector>-Wert nur eine einzelne X, Y oder Z Achse, in dem z.B. der Wert 1 für eine entsprechenden Achse eingetragen wird und die anderen Achsenpositionen den Wert 0 (Null) erhalten.

So benutzt z.B. das `up` Argument des `goto` Kommandos einen <Vector>-Wert, der beschreibt, welche Achse im ausgewählten Koordinatensystem **oben** sein soll. Das nachfolgende Beispiel setzt die Y-Achse nach oben, wobei alle drei <Vector>-Werte (X, Y und Z) eingetragen sein müssen:

```
select { object "Sol/Earth" }
goto   { time 5
         distance 5.5
         upframe "follow"
         up [0 1 0] }
wait   ( duration 5 }
```

<Base64> Dieser Datentyp besteht aus Zeichen, die aus einer Cel://URL *(was eine Cel://URL ist, lesen Sie bitte im weiter oben empfohlenen Benutzerhandbuch nach)* von Celestia kopiert und in Base-64 kodiert wurde.

Beispiel:

Das `setposition` Kommando kann X, Y und Z Werte aus einer Cel://URL nutzen. Die nachfolgenden Werte positionieren die Ansicht außerhalb der Milchstraße:

```
setposition {
  x "AMDCXoJK/3+IyGgR8f///w"
  y "BvP2LRdAAAD/n5UGCw"
  z "VUrGoeQJ+/8DyvenLQ" }
```

Default (Voreinstellung)

Ein Default-Wert (Standard-Wert) wird einem Argument automatisch zugewiesen, wenn Sie dieses Argument nicht im Kommando-Code des Scripts zugewiesen haben.

Aber nicht alle Kommandos und Argumente haben Default-Werte (z.B. `select`). Wenn Sie also kein Argument in Ihrem Script eintragen, stellen Sie sicher, dass es hierzu einen Default-Wert gibt. Andernfalls erhalten Sie einen Scriptfehler.

Beispiel:
Wenn Sie im Kommando `wait` kein `duration` Argument eintragen, wird automatisch der Default-Wert 1.0 zugewiesen.

```
wait {}
```

Anm. d. Übers.: Wenn Sie hingegen dem Kommando `select` kein Argument zuweisen, erhalten Sie einen Scriptfehler:

```
select {}
```

Hier **muss** ein Argument (nämlich `object` mit einem String-Wert) eingetragen werden:

```
select { object "Sol/Earth" }
```

Null

Ein Null-Wert bedeutet, dass keine Argumente für das entsprechende Kommando benutzt werden. So akzeptiert z.B. das Kommando `lock` keinerlei Argumente:

```
lock {}
```

(zwischen den geschweiften Klammern steht nichts)

Anm. d. Übers.: Im weiteren Verlauf dieses Handbuches werden Kommandos, die keine Argumente erfordern schlicht mit „**Argumente: keine**“ gekennzeichnet.

Celestia .CEL Scripts

Was ist ein Celestia .CEL Script?

Ein Celestia .CEL-Script ist eine Datei mit reinem Klartext, die eine Auflistung von Anweisungen für Celestia enthält, die von Celestia automatisch ausgeführt werden.
Das bemerken Sie schon beim Start von Celestia, denn hier wird ein solches Script ausgeführt. Das Start-Script positioniert die Ansicht auf den Jupiter-Mond „Io“ (oder einem anderen Himmelskörper, je nach Version). Dieses Script finden Sie im Hauptverzeichnis von Celestia (`start.cel`).
Vielleicht haben Sie auch schon das Demo abgespielt (durch Drücken der Taste [**D**] oder über das Menü „**Help**“ –> „**Run Demo**“ bzw. „**Hilfe**“ -> „**Demo starten**“). Auch dieses Demo ist ein .CEL-Script und befindet sich ebenfalls im Hauptverzeichnis (`demo.cel`).

Sie können die Scripte mit einem einfachen Text-Editor wie NotePad, WordPad oder Word öffnen. Stellen Sie nur sicher, dass beim Sichern der Datei die Dateierweiterung **.cel** verwendet wird und Sie die Datei als Klartext abspeichern (RTF-Dateien werden z.B. nicht funktionieren). Verwenden Sie daher am besten NotePad.

Wenn Sie nun einen Blick in ein Script werfen möchten, öffnen Sie die Datei **start.cel** im Hauptverzeichnis von Celestia.

	Bevor Sie Modifikationen an Scripten vornehmen, fertigen Sie **vorher** eine Sicherungskopie davon an. Falls erforderlich, können Sie diese Kopie später wieder zurückspielen.

Das **start.cel** Script von Celestia sollte in etwa wie folgt aussehen:

```
{
  select { object "Sol/Jupiter/Io" }
  follow {}
  goto { time 5 }
  # gotolonglat { time 0 distance 1e11 longitude 0 latitude 0 }
  # wait { duration 0.1 }
  # orbit { axis [ 0 1 0 ] rate 10 duration 7 }
  # goto { time 5 distance 10 }
  # wait { duration 5.0 }
}
```

Alle Kommandos und Argumente werden später in diesem Handbuch noch erklärt. Wenn Sie mit den Tastaturbefehlen und Menüpunkten von Celestia bereits vertraut sind, werden Sie schon Vieles wiedererkennen. Das `select` Kommando kommt der Objektauswahl mit **[Enter]** und dem Eintrag eines Objektnamens gleich. Das `follow` Kommando entspricht der Taste [**F**] und das `goto` Kommando hat den gleichen Effekt wie die Taste [**G**].

Die Zeilen mit vorangestelltem „#" sind „auskommentierte" Zeilen, die **nicht** ausgeführt werden. Erst wenn Sie das Doppelkreuz entfernen, wird die entspr. Zeile verarbeitet.
Chris Laurel, der Entwickler von Celestia, hat diese auskommentierten Zeilen eingefügt, damit die Nutzer des Programms damit experimentieren können.

Celestia Script Kommandos schreiben

Das erste Zeichen in einem Celestia .CEL-Script **muss** eine geöffnete geschweifte Klammer sein (**{**). Das letzte Zeichen in einem solchen Script **muss** eine abschließende geschweifte Klammer sein (**}**). Demnach sieht ein leeres, aber gültiges Script wie folgt aus:

```
{ }
```

Sie sehen völlig richtig, die geöffnete und abschließende geschweifte Klammer ist der einzige Inhalt in dieser Datei. Aber natürlich bewirkt dieses Script rein gar nichts, was es ziemlich nutzlos macht. ☺
Ein Script muss also zumindest eine Kommandozeile enthalten, damit es etwas bewirken oder Einstellungen vornehmen kann.

Eine typische einzige Kommandozeile besteht aus den folgenden Bestandteilen:

einem Kommando-Name (wie z.B. `wait`)
einem Leerzeichen
einer geöffneten geschweiften Klammer `{`
keine oder mehrere Argument-Namen (wie z.B. `duration`) mit entspr. Werten
z.B.: `duration 5.5` oder `object "Mars"`
eine abschließende geschweifte Klammer `}`

Nachfolgend einige Kommandozeilen-Beispiele:

Wenn Sie dem Argument `duration` den Wert `5.5` im Kommando `wait` zuweisen, wird Celestia angewiesen, eine Pause von 5,5 Sekunden einzulegen:

```
wait { duration 5.5 }
```

Beachten Sie hierbei, dass **keine Anführungsstriche** (") um den Wert (`5.5`) gesetzt werden dürfen! Anführungsstriche werden ausschließlich bei <String>-Werten verwendet.

Den <String>-Wert `"Mars"` fügen Sie dem Argument `object` im Kommando `select` hinzu:

```
select { object "Mars" }
```

Damit wird Celestia mitgeteilt, dass ein <String>-Wert (Text) übermittelt wird. Ein <String>-Wert muss also immer in Anführungsstrichen gesetzt werden.

Einige .CEL-Script-Kommandos (wie z.B. `print`) senden mehr als nur ein Argument. In einem solchen Fall müssen alle Argumente und die dazugehörigen Werte in geschweiften Klammern gesetzte werden:

```
print { text "Hello universe."  row -4  column 1  duration 5 }
```

oder

```
print { text "Hello universe."
        row -4
        column 1
        duration 5 }
```

Alle Argumente für jedes Kommando müssen in geschweiften Klammern gesetzt werden, wie das obige Beispiel zeigt. Dabei können Sie jedoch Leerzeichen und auch leere Zeilen einfügen, wenn Sie dies für übersichtlicher erachten.

Kommandozeilen

Das Doppelkreuz (#) kann als erstes Zeichen in einer Zeile verwendet werden, um diese damit auszukommentieren (die Zeile wird von Celestia ignoriert und nicht ausgeführt):

```
# ---------------------------------------
#   Das ist mein Celestia Script
# ---------------------------------------
  select { object "Sol/Earth" }
  center { time 3 }
  wait   { duration 3 }
# ---------------------------------------
```

Jede Zeile mit einem Doppelkreuz wird von Celestia ignoriert, nur die drei nicht auskommentierten Zeilen werden verarbeitet. Sie können so z.B. Anmerkungen und Hinweise zu den Kommandos einfügen.

Ihr erstes Celestia .CEL Script

Ihr erstes Script sollte ein Vorlagen-Script sein, welches die immer erforderlichen Vorgaben enthält. So müssen Sie nicht für jedes neue Script immer die gleichen Eingaben vornehmen. So könnte ein (sehr kurzes) Vorlagen-Script aussehen, welches Ihr „Grundgerüst“ für neue Scripts sein kann:

```
{
 #-------------------------------------------------------------------
 # Written by: <Ihr Name>
 #         on: <Datum>
```

```
 #
 # The purpose of this script is ... (Sinn und Zweck des Scripts)
 #-------------------------------------------------------------------
 # Celestia Einstellungen (falls erforderlich)...
 #-------------------------------------------------------------------
 # Folgende Objekte darstellen...
     renderflags { set "cloudmaps|planets|stars" }

 # Markierungen für alle Objekte aufheben...
     unmarkall {}

 # Blickfeld einstellen...
     set { name "FOV" value 30.0 }

 # Zeitablauf einstellen (1x, 100x, 1000x, etc.)...
     timerate { rate 1.0 }
 #-------------------------------------------------------------------
 # Script-Körper (Body) - Codes folgen hier ...
 #-------------------------------------------------------------------

   <Hier kommen Ihre Kommandozeilen hin>

 #-------------------------------------------------------------------
 # Ende des Scripts - Hinweis für die Benutzer ...
 #-------------------------------------------------------------------
   print  { text "The script is finished. / Script beendet."
            row  -3
            column 25
            duration 5 }

   wait   { duration 5 }
}
```

Ein Vorlagen-Script spart Ihnen Zeit, denn Sie müssen sich nicht die Startwerte merken und diese jedes Mal neu eingeben. Außerdem können die Einstellungen und Vorgaben im Vorlagen-Script leicht und schnell geändert werden, wenn Sie ein neues Script schreiben.

Das obige Beispiel-Script demonstriert, wie man die Werte für Kommandozeilen setzt, wie man den Inhalt des Scripts beschreibt, wer das Script geschrieben hat und was jeder Absatz des Scripts veranlasst. Solche Kommentare sind insbesondere dann sinnvoll, wenn Sie Ihr Script mit anderen teilen, die ggf. nicht nachvollziehen können, was und warum Sie etwas mit Ihrem Script veranlassen möchten. Anm. d. Übers.: Sofern Sie dazu in der Lage sind, sollten Sie Ihre Kommentare in englischer Sprache verfassen (es sei denn, Sie schreiben ein Script explizit für deutschsprachige Anwender), denn Celestia ist international verbreitet.

Wenn Sie lieber ein tiefergehendes Vorlagen-Script verwenden möchten, besuchen Sie das ***Celestia Scripting Forum***, das Sie hier finden:

http://www.shatters.net/forum/viewforum.php?f=9.

Ein kostenloses Vorlagen-Script finden Sie dort im folgenden Beitrag:

http://www.shatters.net/forum/viewtopic.php?t=2961 (**Template Script for .cel scripts**).

Oder Sie besuchen ***Don G's Celestia Page*** (http://www.donandcarla.com/Celestia/) und laden sich dort die aktuelle Version des Vorlagen-Scripts herunter.

Script speichern

Wenn Sie ein Celestia .CEL-Script speichern, **muss** die Dateierweiterung (Extension) unbedingt „ **.cel** " (ohne Anführungsstriche!) lauten. Daran erkennt Ihr Betriebssystem, dass diese Datei zu Celestia gehört und Celestia erkennt daran, dass es sich um ein Script handelt.
Außerdem **müssen** Sie den Inhalt der Datei als **reinen Text** abspeichern. RTF-Dateien (Rich Text Format) funktionieren in Celestia nicht, denn derartige Dateiformate enthalten Formatierungen.

Wenn Sie eine Script-Datei doppelklicken, wird das Betriebssystem automatisch Celestia starten (falls nicht bereits geschehen) und Celestia führt anschließend das Script aus.
Anm. d. Übers.: Celestia verhält sich also nicht anders, als z.B. MS Word, wenn Sie im Explorer eine DOC-Datei doppelklicken. Auch hier wird zunächst MS Word gestartet und anschließend das doppelgeklickte Dokument geladen.

Das oben beschriebene Vorlagen-Script können Sie einfach kopieren und in Ihren Text-Editor einfügen und diese Datei dann z.B. unter dem Dateinamen **`template.cel`** oder **`vorlage.cel`** speichern. Oder laden Sie sich das vollständige Vorlagen-Script von ***Don G's Celestia Page*** herunter (URL siehe oben).

Wenn Sie im Folgenden mehr über die Script-Kommandos usw. gelernt haben, werden Sie sicher das Vorlagen-Script modifizieren und es an Ihre eigenen Bedürfnisse anpassen.

Im Moment haben Sie jedoch schon ein fertiges „Anfänger-Script", das Sie für Ihre eigenen Scripts als Vorlage verwenden können. Dazu öffnen Sie einfach Ihre Vorlage, schreiben Ihren Code hinein und speichern dieses neue Script unter einem **neuen** Namen ab.

Koordinatensysteme

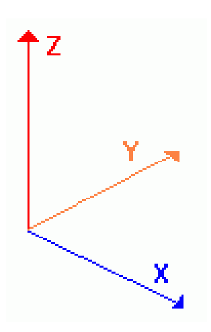
Celestia's Koordinatensysteme

Die nachfolgende Definition wurde aus ***Astronomy Answers -- Astronomical Dictionary*** (http://www.astro.uu.nl/~strous/AA/en/woordenboek.html#coordinate_system) entnommen:

> *Ein Koordinatensystem ist ein Hilfsmittel, das es erlaubt, Positionen durch die Abmessung von Entfernungen in verschiedenen Richtungen zu bestimmen. Diese Entfernungen sind die Koordinaten.*
> *Es gibt dazu zwei oft verwendete Koordinatensysteme: Systeme, die rechteckige (oder kartesische) Koordinaten verwenden und jene, die polare Koordinaten verwenden.*
> *Polare Koordinaten verwenden Winkel (für die Richtung) und eine Entfernungsangabe und haben in der Regel eine flache Basisfläche, an die die Winkel gebunden sind.*
> *Rechteckige Koordinaten verwenden nur die Entfernungen, die in den gegenseitigen orthogonalen (Anm. d. Übers.: also rechtwinkeligen) Richtungen von einem gemeinsamen Punkt gemessen werden.*
> *Koordinaten werden häufig für Objekte genannt, in oder auf die sich die gemessene Position, die Basis oder der Ursprungspunkt bezieht.*
>
> *Die Koordinaten der polaren Systeme, die die Position von Himmelskörpern bestimmen, bestehen oft aus lateinischen oder griechischen Objektnamen, gefolgt von der (geo-)grafischen Lage des Objektes.*
> *Die Namen der entsprechenden rechteckigen Systeme (mit der Mitte des Körpers als Ursprungspunkt) haben normalerweise den gleichen ersten Teil, werden aber gefolgt von der Zentrik (für die Erde Geozentrik), Planetenzentrik im Allgemeinen oder der Heliozentrik für die Sonne.*

Wenn Sie Celestia benutzen, sehen Sie, dass Celestia verschiedene unterschiedliche Modi besitzt, inklusive *Follow*, *Sync Orbit*, *Lock*, *Chase* und *Free* (= keine Bindung, ESC wurde gedrückt).
Zusammen mit einem oder zwei ausgewählten Objekten definieren diese Modi ein Koordinatensystem von Celestia. Im interaktiven Modus können Sie Ihre genaue Position (Länge, Breite, Orientierung/Ausrichtung, Orbit usw.) mit Tastaturbefehle beeinflussen.
Im .CEL-Script gibt es jedoch keine interaktiven Tastaturbefehle, daher müssen Sie Celestia über das Script genau mitteilen, wo Sie sich genau im dreidimensionalen Universum aufhalten wollen und wie die Kamera (also die Ansicht auf dem Monitor) ausgerichtet sein soll. Und genau dazu kommen nun die Koordinatensysteme von Celestia ins Spiel.
Nachfolgend wird also nun beschrieben, wie Sie die X, Y und Z-Achsen des dreidimensionalen Koordinatensystems abstimmen.

Anm. d. Übers.:
Dazu sollten Sie natürlich wissen, welche Achsen welche Richtungen bestimmen:



Ausgehend vom gemeinsamen Mittelpunkt der Achsen (Basis) verläuft hier die X-Achse waagerecht nach rechts, die Z-Achse senkrecht nach oben und die Y-Achse waagerecht nach hinten. Dabei stehen alle Achsen rechtwinkelig (orthogonal) zueinander.
Im Grunde genommen sind diese Koordinaten vergleichbar mit den Angaben zu den Maßen eines Schrankes (Breite x Höhe x Tiefe [BxHxT]). Dabei steht X für die Breite, Z für die Höhe und Y für die Tiefe des Schrankes.
Der Vergleich mit der Bemaßung eines Schrankes ist aber bezogen auf Celestia nicht ganz richtig, denn die Basis befindet sich beim Schrank ja unten in der linken Ecke (und damit außerhalb) des Schrankes. In Celestia, wo jeder Körper kugelförmig ist (Universium, Planeten usw.), befindet sich die Basis im Mittelpunkt (also innerhalb) des Körpers.

Jedes der interaktiven Celestia-Modi wird über das Koordinatensystem im Script mit dem Kommando `setframe` aufgerufen. Die folgende Liste zeigt jeden interaktiven Modus mit dem entsprechenden Koordinatensystem:

Celestia Modus	Koordinatensystem (`setframe`)
Chase	`chase`
Follow	`ecliptical`
Lock	`lock`
Free (keine Bindung)	`universal`
Sync Orbit	`geographic`
--	`equatorial`
--	`observer/local` (wird auch in Lesezeichen verwendet)

Weiter unten finden Sie eine kurze Beschreibung der Koordinatensysteme, die Celestia verwendet. Dies ist ganz hilfreich, wenn Sie die Kommandos `goto`, `gotoloc`, `gotolonglat` und `setframe` verwenden.

Sie können diesen Abschnitt auch zunächst überspringen und ihn erst dann lesen, wenn Sie später tatsächlich in einem Script das Koordinatensystem verwenden müssen.

Alle Koordinaten von Celestia sind orthogonal, das heißt, die X, Y und Z-Achsen sind alle senkrecht miteinander verbunden und alle haben die gleiche Längeneinheit (sind also nicht gestreckt).

`chase` *(Chase)*

Die Chase-Koordinaten (das Script-Kommando heißt genauso) veranlasst Celestia, das aktuell ausgewählten Objekt durch den Weltraum nachzufolgen und dabei die Ausrichtung in der Richtung beizubehalten, in die sich das Objekt bewegt.
Chase ist fast identisch mit **Lock** (siehe weiter unten), nur dass bei **Chase** die Z-Achse in die Richtung zeigt, in die sich das Haupt-Objekt bewegt (also z.B. die Sonne für die Planeten oder ein Planet für dessen Monde). Der Tastaturbefehl für **Chase** ist **["]**.

Im Chase-Modus bleibt Ihre Position also fixiert auf die relative Bewegung des ausgewählten Objektes. Wenn Sie sich direkt vor oder hinter dem Ziel befinden (je nach dessen Bewegung) verbleiben Sie auch dort, egal wohin sich das Objekt tatsächlich bewegt.

Wenden Sie den Chase-Modus mal auf einen Kometen (aber nicht Halley) an.
Weil Kometen im Allgemeinen sehr exzentrische Bahnen haben, ist deren Bewegungsrichtung nicht immer senkrecht zur Bewegungsrichtung der Sonne. Im Lock-Modus wird die Sonne fixiert bleiben, im Chase-Modus jedoch wird sie durch das Blickfeld wandern.

`ecliptical` *(Follow)*

Das ekliptische Koordinatensystem veranlasst Celestia, dem ausgewählten Objekt zu folgen: Wenn das Objekt sich durch den Weltraum bewegt, bewegt sich die Kamera (die Ansicht) mit ihm. Dabei dreht sich das Objekt um seine eigene Achse in der Ansicht (die Längengrade auf dem Objekt ändern sich stetig). Die Ansicht verweilt jedoch in gleichbleibendem Abstand und auf gleichem Breitengrad über dem Objekt. Der Tastaturbefehl für **Follow** ist **[F]**.
Die X-Achse zeigt von der Sonne weg in Richtung des julianischen 2000.0 Frühlingsäquinoktikums. Die Y-Achse verhält sich dabei normal zur Ekliptik mit positivem Nordgrad. Die Z-Achse komplettiert das Koordinatensystem.

`equatorial` *(ohne interaktivem Befehl)*

Dieses Koordinatensystem ist nur über das Kommando `setframe` aufrufbar, jedoch nicht als Tastatur- oder Menübefehl.

`lock` *(Lock)*

Dieses Koordinatensystem bezieht zwei Objekte ein. Das zuerst gewählte Objekt nennt sich **Referenzobjekt**, das zweite (anschließend) ausgewählte nennt sich **Zielobjekt**.
Wenn zwei Objekte als ein Paar miteinander verbunden werden und Sie diese Szenerie umkreisen, bleibt das Zielobjekt am Referenzobjekt angebunden. Die oben links auf dem Monitor angezeigte Entfernung ist der Abstand zwischen diesen beiden Objekten (und nicht wie normalerweise üblich, der Abstand zwischen Ihnen und dem Objekt). Der Tastaturbefehl für Lock ist **[:]**.

Die Z-Achse reicht vom Referenzobjekt zum Zielobjekt. Die Y-Achse liegt auf der Ebene im rechten Winkel zwischen der Z-Achse und der Rotationsachse des Referenzobjektes.

Im Lock-Modus verbleibt Ihre Position auf einer relativen Linie zwischen dem Referenzobjekt und dem Zielobjekt. Wenn das Zielobjekt sich um das Referenzobjekt bewegt, so werden Sie sich ebenso um das Referenzobjekt bewegen.
Wenn das Zielobjekt also z.B. die Sonne ist, bleibt der beleuchtete Teil eines Planeten (seine „Phase“) immer gleich, denn Sie bewegen sich synchron mit der Sonne um diesen Planeten.

	**Hinweis der Entwickler:** Leider funktioniert diese Definition des Koordinatensystems nicht, wenn die Rotationsachse perfekt auf die Richtung des Referenzobjektes zum Zielobjekt ausgerichtet ist. Außerdem produziert dieses Koordinatensystem merkwürdige Resultate bei einigen Körpern. Dies soll zukünftig dahingehend geändert werden, dass die Definition nicht mehr über die Rotationsachse bestimmt wird.

`universal` *(keine Bindung)*

Wenn das Koordinatensystem in den universellen Modus geschaltet wird, werden alle zuvor ausgewählten Koordinatensysteme abgewählt (abgeschaltet). Den gleichen Effekt erzielen Sie durch Drücken der Taste **[Esc]**. Alle o.g. Bindungen zum Objekt (Follow, Lock, Chase usw.) werden also aufgehoben und das Objekt schwebt oder driftet frei im Weltraum.

`Observer/Local` *(ohne interaktivem Befehl)*

Dieses Koordinatensystem ist nur über das Kommando `setframe` aufrufbar, jedoch nicht als Tastatur- oder Menübefehl.

`geographic` *(Sync Orbit)*

Das geografische Koordinatensystem erlaubt Ihnen, stationär bzw. geostationär (auch „geosynchron“ genannt) über dem gewählten Objekt in dessen Orbit zu verweilen. Diese Bindung funktioniert nicht nur bei der Erde, sondern auch bei anderen Objekten.

Wenn das Objekt unter Ihnen um seine eigene Achse rotiert, bewegen Sie sich synchron mit dieser Umdrehung (Sie bleiben also immer an der gleichen Stelle über dem Objekt). Der Tastaturbefehl für **SyncOrbit** ist **[Y]**.
Hierbei rotiert die Achse *mit* dem ausgewählten Objekt. Die Y-Achse ist die Rotationsachse im Uhrzeigersinn, sie zeigt also nach Norden bei rechtläufigen (porgraden) Körpern (wie z.B. die Erde) und nach Süden bei gegenläufigen (retrograden) Körpern (wie z.B. die Venus). Die X-Achse zeigt vom Zentrum eines Objektes zum Durchschnitt seines Längenmeridians und Äquator. Die Z-Achse (rechtwinkelig zur XY-Fläche) komplettiert dieses Koordinatensystem.

	Der **Ursprung** (die Basis) aller Koordinatensysteme von Celestia (außer `universal`) ist die Mitte der **Referenzobjekte**. Wenn Sie beispielsweise Merkur folgen (follow) und das Kommando `gotoloc` mit der Positionsargument `position [0 0 0]` ausführen, fliegen zum Mittelpunkt von Merkur. Im Universal-Mode `universal [0 0 0]` fliegen Sie dagegen in die Nähe der Sonne. Das Argument `position` im Kommando `gotoloc` ist ein Punkt im aktuell aktiven Koordinatensystem (Maßeinheit: km).

Mehr Informationen zu Koordinatensysteme:

Astronomy Answers -- Astronomical Dictionary (nur Text):
http://www.astro.uu.nl/~strous/AA/en/woordenboek.html#coordinate_system

Coordinate Systems Overview (Text und Grafiken):
http://www.colorado.edu/geography/gcraft/notes/coordsys/coordsys_f.html

Celestial Coordinate System (Text und Grafiken):
http://csep10.phys.utk.edu/astr161/lect/time/coordinates.html

Coordinate Systems in Astronomy (Text und Grafiken):
http://www.astro.virginia.edu/~teacha/130_manual/node9.html

Earth Coordinate System (Text und Grafiken):
http://zebu.uoregon.edu/~js/ast121/lectures/lec02.html

Beyond Our Skies: Discovering the Cosmos -- Coordinate Systems (Text und Grafiken):
http://library.thinkquest.org/29033/begin/coordinate.htm

Coordinate system transformation (Gleichungen und Grafiken):
http://astronomy.swin.edu.au/~pbourke/projection/coords/

.CEL Script Kommandos

Beschreibung der .CEL-Script Kommandos

Dieser Abschnitt listet alle Celestia .CEL-Script-Kommandos für die Version 1.3.1 auf. Dazu finden Sie eine Beschreibung, was jedes Kommando veranlasst, welche Argumente es hat und wie Sie die Kommandos kodieren.

Übrigens ist das Experimentieren der erfreulichste Teil des Lernens. Nehmen Sie sich also die Zeit und experimentieren Sie mit den Kommandos. Wenn Sie dabei einen besonders interessanten Effekt entdecken, sollten Sie die Codierung (also die Kommandozeilen) mit den anderen Celestia-Nutzern teilen und Ihren Code im ***Celestia Scripting Forum*** (http://www.shatters.net/forum/viewforum.php?f=9) bereitstellen.

Der erste Teil dieses Abschnittes liefert eine alphabetische Referenzliste der Kommandos und eine nach Kommandotypen kategorisierte Liste.

	* **Alle Kommandos müssen in Kleinbuchstaben eingegeben werden**, sonst werden sie von Celestia nicht erkannt. Sie sollten auch alle Argumente in Kleinbuchstaben eingeben, obwohl dies nicht zwingend erforderlich ist. * Wenn Sie ein Kommando verwenden, das ein `time` oder `duration` Argument enthält (z.B. `goto`, `print`, `center` usw.), **müssen** Sie danach das Kommando `wait` mit einer Dauer (`duration`) einfügen, die gleich oder länger als die Zeitangabe im Kommando ist. Wenn Sie dies vergessen, wird das Kommando entweder ignoriert oder es funktioniert nicht so, wie Sie es vorgesehen haben.

Alphabetische Liste der Kommandos

`cancel`	Hebt das `goto` und `track` Kommando auf und setzt das Koordinatensystem zurück auf `universal`.
`center`	Zentriert das ausgewählte Objekt auf dem Monitor.
`changedistance`	Ändert den Abstand zum ausgewählten Objekt.
`chase`	Setzt das Koordinatensystem auf `chase`.
`cls`	Löscht alle Bildschirminhalte, die mit dem Kommando `print` aufgerufen wurden.
`follow`	Folgt dem ausgewählten Objekt und setzt das Koordinatensystem auf `ecliptical`.
`goto`	Geht zum ausgewählten Objekt und wendet das aktuell aktive Koordinatensystem an.
`gotoloc`	Geht unter Verwendung des aktiven Koordinatensystems zur angegebenen Position und Ausrichtung des ausgewählten Objektes.

gotolonglat	Geht zum angegebenen longitude (Längengrad) und latitude (Breitengrad) des ausgewählten Objektes.
labels	Schaltet die labels (Bezeichnungen) an oder ab.
lock	Verbindet (koppelt) zwei Objekte in der Ansicht und setzt das Koordinatensystem auf lock.
lookback	Ändert die Ansicht um 180° (=dreht die Blickrichtung um).
mark	Markiert das definierte Objekt mit einem Symbol.
move	Bewegt die Ansicht mit der angegebenen Geschwindigkeit.
orbit	Umläuft das ausgewählte Objekt unter Verwendung des aktiven Koordinatensystems.
preloadtex	Lädt eine Textur-Datei vorab in den Speicher.
print	Schreibt eine Text-Mitteilung auf den Bildschirm.
renderflags	Schaltet bestimmte Einstellungen (z.B. Wolken, Atmosphäre usw.) ein oder aus.
rotate	Dreht die Ansicht (Kamera).
select	Wählt ein Objekt aus (Planet, Mond, Galaxie usw.).
set	Bestimmt die minimale Orbitgröße (Min Orbit Size), die Stufe des Streulichtes (Ambient Light Level), das Ausmaß des Blickfeldes (Field Of View), das Limit der Entfernungen für Sterne (Star Distance Limit) oder das Aussehen der Sterne (Star Style).
setfaintestautomag45deg	Bestimmt die Magnitude (scheinbare Helligkeit) der darzustellenden Sterne, wenn der Modus “AutoMag” **ein**geschaltet ist.
setframe	Bestimmt das zu verwendende Koordinatensystem.
setorientation	Stellt die Lage/Orientierung der Ansicht (Kamera) ein.
setposition	Stellt die Position der Ansicht im Weltraum ein.
setsurface	Stellt den Namen der alternativen Oberfläche für das ausgewählte Objekt ein.
setvisibilitylimit	Bestimmt die Magnitude (scheinbare Helligkeit) der darzustellenden Sterne, wenn der Modus “AutoMag” **aus**geschaltet ist.
seturl	Bewegt die Ansicht (Kamera) zu dem Ort einer gespeicherten “location URL” (Cel://URL).
synchronus	Schaltet den Modus “Sync Orbit” für das ausgewählte Objekt ein und setzt das Koordinatensystem auf geographic.
time	Stellt das Datum und die Uhrzeit ein.
timerate	Stellt den Faktor für den Zeitmultiplikator ein (z.B. x10, x100, x1000 usw.)
track	Hält das gewählte Objekt in der Bildschirmmitte.
unmark	Entfernt die Markierung für das angegebene Objekt.

`unmarkall`	Entfernt die Markierungen für alle Objekte und schaltet die Anzeige der Markierungen ab.
`wait`	Hält das Script für die angegebene Dauer (in Sekunden) an.

Kommandos (sortiert nach Kategorien): *Objektkontrolle und Anzeige*

`center`	Zentriert das ausgewählte Objekt auf dem Monitor.
`chase`	Setzt das Koordinatensystem auf `chase`.
`follow`	Folgt dem ausgewählten Objekt und setzt das Koordinatensystem auf `ecliptical`.
`lock`	Verbindet (koppelt) zwei Objekte in der Ansicht und setzt das Koordinatensystem auf `lock`.
`lookback`	Ändert die Ansicht um 180° (=dreht die Blickrichtung um).
`orbit`	Umläuft das ausgewählte Objekt unter Verwendung des aktiven Koordinatensystems.
`rotate`	Dreht die Ansicht (Kamera).
`select`	Wählt ein Objekt aus (Planet, Mond, Galaxie usw.).
`setframe`	Bestimmt das zu verwendende Koordinatensystem.
`setorientation`	Stellt die Lage/Orientierung der Ansicht (Kamera) ein.
`setposition`	Stellt die Position der Ansicht im Weltraum ein.
`setsurface`	Stellt den Namen der alternativen Oberfläche für das ausgewählte Objekt ein.
`seturl`	Bewegt die Ansicht (Kamera) zu dem Ort einer gespeicherten “location URL” (Cel://URL).
`synchronus`	Schaltet den Modus “Sync Orbit” für das ausgewählte Objekt ein und setzt das Koordinatensystem auf `geographic`.
`track`	Hält das gewählte Objekt in der Bildschirmmitte.

Kommandos (sortiert nach Kategorien): *Gesamtansicht u. Rendering-Flags*

`cls`	Löscht alle Bildschirminhalte, die mit dem Kommando `print` aufgerufen wurden.
`labels`	Schaltet die `labels` (Bezeichnungen) an oder ab.
`lookback`	Ändert die Ansicht um 180° (=dreht die Blickrichtung um).
`mark`	Markiert das definierte Objekt mit einem Symbol.
`preloadtex`	Lädt eine Textur-Datei vorab in den Speicher.
`print`	Schreibt eine Text-Mitteilung auf den Bildschirm.
`renderflags`	Schaltet bestimmte Einstellungen (z.B. Wolken, Atmosphäre usw.) ein oder aus.
`set`	Bestimmt die minimale Orbitgröße (Min Orbit Size), die Stufe des Streulichtes (Ambient Light Level), das Ausmaß des Blickfeldes (Field Of View), das Limit der Entfernungen für Sterne (Star Distance Limit) oder das Aussehen der Sterne (Star Style).
`setfaintestautomag45deg`	Bestimmt die Magnitude (scheinbare Helligkeit) der darzustellenden Sterne, wenn der Modus "AutoMag" **ein**geschaltet ist.
`setframe`	Bestimmt das zu verwendende Koordinatensystem.
`setsurface`	Stellt den Namen der alternativen Oberfläche für das ausgewählte Objekt ein.
`setvisibilitylimit`	Bestimmt die Magnitude (scheinbare Helligkeit) der darzustellenden Sterne, wenn der Modus "AutoMag" **aus**geschaltet ist.
`unmark`	Entfernt die Markierung für das angegebene Objekt.
`unmarkall`	Entfernt die Markierungen für alle Objekte und schaltet die Anzeige der Markierungen ab.

Kommandos (sortiert nach Kategorien): *Bewegungen und Navigation*

`cancel`	Hebt das `goto` und `track` Kommando auf und setzt das Koordinatensystem zurück auf `universal`.
`center`	Zentriert das ausgewählte Objekt auf dem Monitor.
`changedistance`	Ändert den Abstand zum ausgewählten Objekt.
`chase`	Setzt das Koordinatensystem auf `chase`.
`follow`	Folgt dem ausgewählten Objekt und setzt das Koordinatensystem auf `ecliptical`.
`goto`	Geht zum ausgewählten Objekt und wendet das aktuell aktive Koordinatensystem an.
`gotoloc`	Geht unter Verwendung des aktiven Koordinatensystems zur angegebenen Position und Ausrichtung des ausgewählten Objektes.
`gotolonglat`	Geht zum angegebenen `longitude` (Längengrad) und `latitude` (Breitengrad) des ausgewählten Objektes.
`lookback`	Ändert die Ansicht um 180° (=dreht die Blickrichtung um).
`move`	Bewegt die Ansicht mit der angegebenen Geschwindigkeit.
`orbit`	Umläuft das ausgewählte Objekt unter Verwendung des aktiven Koordinatensystems.
`rotate`	Dreht die Ansicht (Kamera).
`setframe`	Bestimmt das zu verwendende Koordinatensystem.
`setorientation`	Stellt die Lage/Orientierung der Ansicht ein.
`setposition`	Stellt die Position der Ansicht im Weltraum ein.
`seturl`	Bewegt die Ansicht (Kamera) zu dem Ort einer gespeicherten “location URL” (Cel://URL).
`synchronus`	Schaltet den Modus “Sync Orbit” für das ausgewählte Objekt ein und setzt das Koordinatensystem auf `geographic`.
`track`	Hält das gewählte Objekt in der Bildschirmmitte.

Kommandos (sortiert nach Kategorien): *Datum und Zeit*

`time`	Stellt das Datum und die Uhrzeit ein.
`timerate`	Stellt den Faktor für den Zeitmultiplikator ein (z.B. x10, x100, x1000 usw.)
`wait`	Hält das Script für die angegebene Dauer (in Sekunden) an.

Alphabetische Liste der Kommandos (mit Beispielen)

---

cancel

Argumente: keine

Hebt das goto und track Kommando auf und setzt das Koordinatensystem auf universal zurück. Das bedeutet, dass auch die Kommandos follow, synchronous und andere sich auf Koordinaten beziehende Kommandos abgeschaltet werden. Somit entspricht das Kommando cancel dem Druck auf die Taste **[Esc]** in der interaktiven Benutzung des Programms.

Wenn Sie von einem Objekt zum nächsten springen, sollten Sie das Kommando cancel zwischen diesen beiden Objekten einfügen, damit evtl. vorhandene Bindungen des aktuell aktiven Objekts aufgehoben werden. Wenn Sie das aktuelle Koordinatensystem beibehalten wollen (falls es ein anderes als universal sein sollte), müssen Sie es mit setframe nach der Ausführung von cancel wiederherstellen.

Beispiel:
Der erste Teil des nachfolgenden Scripts wählt die Erde (Earth) aus, geht dort hin und führt dann weitere Kommandos aus. Der zweite Abschnitt führt ein cancel Kommando aus, wählt dann Mars und führt weitere Kommandos aus:

```
  select { object "Sol/Earth" }
  goto   { time 3 }
  wait   ( duration 3 }
# ...
# <weitere Kommandos>
# ...
  cancel { }
  select { object "Sol/Mars" }
  goto   { time 3 }
  wait   { duration 3 }
# ...
# <weitere Kommandos>
# ...
```

Wenn das aktuell gewählte Objekt mit follow belegt ist, dann wird es ***automatisch*** durch die Auswahl eines neuen Objektes (select) mit dem follow Kommando überschrieben.

Wenn das aktuell gewählte Objekt mit track belegt ist, können Sie nun seit Celestia 1.3.1 mit dem nachfolgenden Code ausschließlich das track Kommando aufheben:

```
select { object "" }
track  { }
```

center

Argument: `time <Wert>` (Voreinstellung: 1.0)

Zentriert das ausgewählte Objekt in die Mitte des Bildschirms. Sie müssen zunächst das Kommando `select` verwenden, bevor Sie `center` anwenden können. Wenn Sie das `time` Argument nicht zuweisen, wird die Voreinstellung (1 Sekunde) benutzt.

`time`	Wert in Sekunden, bis das Objekt zentriert wird.

	***Verwenden Sie keinen negativen Wert für dieses Argument! Andernfalls hängt sich Celestia auf.***

Das nachfolgende Beispiel wählt die Erde aus und benötigt zum Zentrieren der Erde 5,5 Sekunden. Sollten Sie die Erde nicht sehen können (weil Ihre aktuelle Position viel zu weit von der Erde entfernt ist), drücken Sie die Taste **[G]** und die Erde wird herangezoomt:

```
select { object "Sol/Earth" }
center { time 5.5 }
```

changedistance

Argumente:
`duration <Wert>` (Voreinstellung: 1.0)
`rate     <Wert>` (Voreinstellung: 0.0)

Ändert die Entfernung der Ansicht (Kamera) zum ausgewählten Objekt. Sie müssen auch hier zunächst das `select` Kommando voranstellen.

`duration`	Wert in Sekunden, bis die Entfernung geändert wird.
`rate`	Geschwindigkeit, mit der die Änderung der Entfernung ausgeführt wird. Kleine Dezimalstellen-Werte funktionieren am besten (5 – 6). Ein negativer Wert bewegt die Ansicht näher an das Objekt, während ein positiver Wert die Ansicht vom Objekt entfernt (ein Pluszeichen ist dazu nicht erforderlich).

	***Verwenden Sie keine negative Werte für das Argument `duration`, sonst wird dieses Argument ignoriert.***

Das nachfolgende Beispiel wählt die Erde aus, begibt sich dort hin und benötigt anschließend 2,5 Sekunden, um die Entfernung zu ändern:

```
select { object "Sol/Earth" }
goto   { time 3 }
wait   { duration 3 }
changedistance { duration 2.5 rate 0.5 }
wait   { duration 2.5 }
```

chase

Argumente: keine

Veranlasst Celestia das aktive Koordinatensystem auf `chase` zu setzen und das ausgewählte Objekt durch das Weltall zu verfolgen. Damit verbleibt die Ausrichtung zur Fluchtrichtung des Objektes konstant. Das Kommando ähnelt dem Kommando `lock` (siehe weiter unten), allerdings zeigt die Z-Achse in die relative Bewegungsrichtung des Referenzobjektes.

Im Chase-Modus bleibt Ihre Position also fixiert auf die relative Bewegung des ausgewählten Objektes. Wenn Sie sich direkt vor oder hinter dem Ziel befinden (je nach dessen Bewegung) verbleiben Sie auch dort, egal wohin sich das Objekt tatsächlich bewegt.

Bevor Sie das `chase` Kommando anwenden, müssen Sie erst ein Objekt mit dem `select` Kommando bestimmen.



Es kann immer nur ein Koordinatensystem zur gleichen Zeit aktiv sein:

- `chase` (Chase)
- `equatorial`
- `ecliptical` (Follow)
- `geographic` (Sync Orbit)
- `lock` (Lock)
- `observer`
- `universal` (ohne Bindungen)

Das folgende Beispiel wählt den Mond aus, setzt das Koordinatensystem auf `chase` und bringt Sie anschließend zum Mond:

```
select { object "Sol/Earth/Moon" }
chase  { }
goto   { time 2 }
wait   { duration 2 }
```

cls

Argumente: keine

Löscht alle Textinformationen auf dem Bildschirm, die vorher mit dem Kommando `print` aufgerufen wurden.
Eigentlich benötigen Sie dieses Kommando nicht, denn mit einem erneuten `print` Kommando wird der Text sowieso überschrieben. Meist werden Sie auch den anzuzeigenden Text mit einer Anzeigedauer versehen, nach der der Text entfernt wird.

Beispiel:

```
cls { }
```

follow

Argumente: keine

Dieses Kommando setzt das Koordinatensystem auf `ecliptical` und folgt dem ausgewählten Objekt.

Sie müssen auch hier zunächst ein Objekt mit `select` auswählen.




Es kann immer nur ein Koordinatensystem zur gleichen Zeit aktiv sein:

- `chase` (Chase)
- `equatorial`
- `ecliptical` (Follow)
- `geographic` (Sync Orbit)
- `lock` (Lock)
- `observer`
- `universal` (ohne Bindungen)

Wenn Sie `follow` auf die Erde anwenden, wird die Erde im Prinzip zum Zentrum des Sonnensystems, in dem die Sonne sich um die Erde dreht.

Das Beispiel wählt den Mars aus, setzt das Koordinatensystem auf `ecliptical` (`follow`) und bringt Sie zum Mars:

```
select { object "Sol/Mars" }
follow { }
goto   { time 2 }
wait   { duration 2 }
```

goto

Argumente:

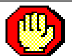
```
time     <Wert>    (Voreinstellung: 1.0)
distance <Wert>    (Voreinstellung: 5.0)
upframe  <String>  (Voreinstellung:  "observer")
up       <Vector>  (Voreinstellung:  [0 1 0] "Y")
```

Bewegt die Ansicht (Kamera) unter Verwendung des angegebenen Koordinatensystems (`upframe`) in der angegebenen Zeit (`time`) zum ausgewählten Objekt, stoppt in der angegebenen Entfernung (`distance`) und setzt die Achse des Objektes senkrecht (`up`).

Auch hier müssen Sie zuvor das Objekt mit dem Kommando `select` auswählen.

`time`	Zeitraum in Sekunden, in dem das Objekt angefahren wird.

distance	Gibt an, wie weit weg Sie sich von dem Objekt positionieren möchten. Die Maßeinheit hierfür beträgt: Objektradius plus 1. Wenn Sie als Wert für distance 6 angeben und der Radius des Objektes 10.000 km beträgt, werden Sie 50.000 km vom Mittelpunkt des Objektes entfernt positioniert. (Anm. d. Übers.: Klingt komisch, is' aber so! ☺) Am einfachsten berechnen Sie den erforderlichen Wert wie folgt: Gewünschte Entfernung geteilt durch Objektradius plus 1 Nehmen wir das Beispiel von oben: Sie möchten 50.000 km von einem Objekt mit einem Radius von 10.000 km entfernt positioniert werden. Dann sieht die Rechnung wie folgt aus: (50.000 / 10.000) + 1 = **6** Anderes Beispiel mit der Erde: Sie möchten 1.000.000 km von der Erde entfernt positioniert werden. Die Erde hat einen Radius von 6.378,1 km. Dann sieht die Rechnung wie folgt aus: (1000000 / 6,378.1) + 1 = **157.7865** **Vorsicht** mit folgenden Werten für distance : 0 (Null) = Mittelpunkt des Objektes. Wenn Sie diesen Wert in der Version 1.3.1 verwenden, kann Celestia anschließend weitere Positionierungen nicht mehr richtig erkennen! Sie sollten deshalb den Wert „Null“ nicht einsetzen. 1 = Oberfläche des Objektes. Wenn Sie die Ansicht ganz genau auf die Oberfläche des Objektes herabsenken (Höhe = 0 km), zeigen einige Grafikkarten nur noch Polygone an. Am besten setzen Sie daher einen Wert etwas **oberhalb** der Oberfläche ein.
upframe	Aktiviert das angegebene Koordinatensystem, welches eines der nachfolgenden Werte aufweisen **muss**: chase ecliptical equatorial geographic lock observer universal
up	Definiert, welche Achse nach oben zeigt. X [1 0 0], Y [0 1 0] oder Z [0 0 1]

	***Verwenden Sie keinen negativen Wert für das time Argument! Andernfalls hängt sich Celestia auf.***

	* Sie können andere Kommandos benutzen, die keine Bewegungen ausführen (z.B. print), während sich die Ansicht zum Ziel begibt. * Bevor Sie andere Kommandos mit einem time Argument benutzen, **müssen** Sie erst ein wait Kommando ausführen lassen, welches die gleiche oder eine längere Dauer als das time Argument des goto Kommandos haben muss. Andernfalls wird das goto Kommando nicht ausgeführt, stattdessen wird das nächste Kommando mit einem time Argument ausgeführt.

Das folgende Beispiel wählt den Mars aus und benötigt 5 Sekunden zum Hinfliegen:

```
select { object "Mars" }
goto   { time 5 }
wait   { duration 5 }
```

Ungültiges Beispiel:
Im Folgenden ein Beispiel, wie man ein zeitbasierendes Kommando **nicht** einrichten sollte. Man könnte hierbei meinen, dass die Serie der Kommandos den Mars auswählen (select) würde und 5 Sekunden zum Hinfliegen benötigt, dann die Erde ausgewählt und 3 Sekunden zum Hinfliegen benötigt würde:

```
select { object "Mars" }
goto   { time 5 }
select { object "Earth" }
goto   { time 3 }
wait   { duration 8 }
```

Da aber das wait Kommando nach dem **ersten** goto Kommando ***nicht*** eingebunden wurde, ignoriert Celestia das **erste** goto Kommando und führt nur das **zweite** goto Kommando aus. Sie würden also nur die Erde sehen, nicht aber den Mars.

Anm. d. Übers.: Diesen Umstand sollten Sie sich gut merken, denn ein fehlendes wait Kommando nach time Argumenten führt **immer** dazu, dass ausgewählte Objekte beim Ausführen des Scripts nicht in Erscheinung treten und man sich wundert, warum dies so ist. Im obigen Beispiel hat Celestia gar keine Zeit, den Mars auszuwählen und in der angegebene Zeit dort hinzufliegen, denn bevor diese Kommandos überhaupt ausgeführt werden können, erhält Celestia schon die Anweisung, die Erde auszuwählen. Mit dem wait Kommando müssen Sie Celestia also quasi anweisen, zunächst abzuwarten, bis die davorstehenden Kommandos abgearbeitet sind.

Das gültige Script **muss** also wie folgt aussehen:

```
select { object "Mars" }
goto   { time 5 }
wait   { duration 5 } ←
select { object "Earth" }
goto   { time 3 }
wait   { duration 3 }
```

Im nächsten Beispiel wird gezeigt, wie man ein weiteres Kommando (das nicht zeitbasiert ist) nach dem `goto` Kommando einbinden kann. Hier wird eine Textmitteilung angezeigt, während Celestia zum Mars fliegt:

```
select { object "Mars" }
goto   { time 5
         distance 8.5
         upframe "follow"
         up [0 1 0] }
print  { text "Wir sind auf dem Weg zum Mars."
         row -3 column 1
         duration 5 }
wait   { duration 5 }
```

---

gotoloc

Argumente:

```
time     <Wert>      (Voreinstellung: 1.0)
position <Vector>    (Voreinstellung: [0 1 0])
xrot     <Wert>      (Voreinstellung: 0.0)
yrot     <Wert>      (Voreinstellung: 0.0)
zrot     <Wert>      (Voreinstellung: 0.0)
```

-oder-

```
time     <Wert>
x        <Base64>
y        <Base64>
z        <Base64>
ow       <Wert>
ox       <Wert>
oy       <Wert>
oz       <Wert>
```

Das `gotoloc` Kommando bewegt die Ansicht in der angegeben Zeit zum aktuell ausgewählten Objekt und begibt sich in die angegebene Position (bzw. zu den Base64-Werten `x`, `y` und `z` aus einer Cel:/URL). Zur Ausrichtung werden die über `xrot`, `yrot` und `zrot` (bzw. `ow`, `ox` und `oz`) angegebenen Werte benutzt.

Auch hier ist zunächst die Auswahl eines Objektes mit `select` erforderlich.

Beachten Sie, dass `gotoloc` nicht das aktuelle Koordinatensystem verändert, sondern das eingestellte Koordinatensystem beibehält.

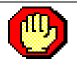
Wenn Sie eine Position aus einer Cel:/URL duplizieren möchten, müssen Sie also das richtige Koordinatensystem, die Position und andere Parameter einstellen (siehe auch `seturl`).

Die X, Y und Z-Werte repräsentieren die Position, wie sie von einer Cel://URL gespeichert wird. Die OW, OX und OZ-Werte repräsentieren die

Ausrichtung/Orientierung (xrot, yrot und zrot), wie sie von einer Cel://URL gespeichert wird. Diese Werte werden wie folgt aus Cel:/URLs übergeben:

&x= &y= &z= und &ow= &ox= &oy= &oz=

time	Zeitraum in Sekunden, in dem das Objekt angefahren wird.
position	Definiert einen Punkt im aktuellen Koordinatensystem in der Maßeinheit Kilometer. Außer im universal Koordinatensystem bestimmt die position [0 0 0] in allen anderen Koordinatensystemen den Mittelpunkt des Objektes. Der erste Wert stellt den Ort der Ansicht in Kilometer entlang der X-Achse dar. Der zweite Wert bezieht sich auf die Y-Achse und der dritte Wert auf die Z-Achse (alle Werte in Kilometer). Der Kilometer-Wert muss den Radius des Objektes beinhalten. Wenn Sie also beispielsweise 100.000 km über der Oberfläche eines Objektes mit einem Radius von 6.378,1 km (Erde) schweben möchten, müssen Sie den Radius zu Ihrer gewünschten Höhe von 100.000 km hinzuaddieren, was einen Wert von 106378.1 ergibt.
xrot yrot zrot	xrot, yrot und zrot sind „Eular Winkel“ oder „Winkel-Achsen“, die die Orientierung/Ausrichtung der Ansicht repräsentieren. Stellen Sie sich diese als Höhenwinkel, Lenkwinkel und Rollwinkel in der Luftfahrt vor. Diese Werte werden also benutzt, um die Blickrichtung von Celestia festzulegen. Die entsprechenden interaktiven Tastaturbefehle lauten wie folgt: X/Pitch (Pfeil nach Oben / Pfeil nach Unten) Y/Yaw (auf Nummernblock: 4 und 6) Z/Roll (Pfeil nach Links / Pfeil nach Rechts)

	***Verwenden Sie keinen negativen Wert für das time Argument! Andernfalls hängt sich Celestia auf.***

Das folgende Beispiel stellt die Erde in 100.000 km Entfernung dar:

```
select     { object "Sol/Earth" }
center     { }
follow     { }

gotoloc    { time 2
             position [0 106378.1 0]
             xrot 90
             yrot 0
             zrot 0 }
wait       { duration 2 }
```

Das nächste Beispiel ist ein wenig komplexer und demonstriert beide `gotoloc` Methoden. Es fliegt Sie zur Erde, setzt die Position so, dass Sie die Nachtlichter über den USA sehen können und stellt die Sonne in der oberen linke Ecke des Monitors dar, während der Mond sich in der oberen rechten Ecke befindet.

(Anm. d. Übers.: Wichtig in diesem Zusammenhang sind die **fettgedruckten** Zeilen der `gotoloc` Kommandos weiter unten im Script):

```
{
  cancel { }

  renderflags { clear "boundaries|comettails|constellations" }
  renderflags { clear "eclipseshadows|orbitspointstars|ringshadows" }
  renderflags { clear "automag|grid" }
  renderflags { set   "cloudmaps|galaxies|nightmaps|planets|stars" }
  renderflags { set   "atmospheres|markers" }

  labels     { clear "planets|moons|spacecraft|asteroids" }
  labels     { clear "constellations|stars|galaxies" }

# Pause time and set the date and time...
  timerate   { rate 0 }
  time       { jd 2452874.692638889 }

# Place a blue square marker around the Moon...
  unmarkall { }
  mark       { object "Sol/Earth/Moon"
               size   20.0
               color  [0 0 1]
               symbol "square" }

# Set the Field Of View...
  set        { name "FOV" value 47.0 }

  select     { object "Sol/Earth" }
  center     { }
  follow     { }

# Goto location: +7000km-X, +9000km-Y, +11000km-Z
# Pitch Up +13X, Yaw Left -32Y, Roll Left -95Z
  gotoloc    { time 2
               position [7000 9000 11000]
               xrot 13
               yrot -32
               zrot -95 }
  wait       { duration 2 }

  print      { text "Blue square is the Moon ---"
               row -24 column 32 duration 3 }
  wait       { duration 3 }

#*******************************************
  print { text "First result done."
          duration 2  row -3  column  2 }
  wait { duration 2 }
```

```
#****************************************

  cancel { }
  gotoloc {
    time 2
    x "wJjebddBM3XLDA"
    y "Z2/b0Q34Pw"
    z "GShiyVOKuxUI"
    ow 0.954531
    ox 0.112769
    oy -0.272262
    oz -0.045019 }

#****************************************
  print { text "Second result done."
          duration 2  row -3  column  2 }
  wait { duration 2 }
#****************************************

}
```

Anm. d. Übers. : Die in den v.g. Script-Beispielen enthaltenen kyrptisch anmutenden Werte scheinen auf den ersten Blick kaum nachvollziehbar und schwer verständlich. Wenn Sie jedoch ein wenig damit experimentieren und Werte aus Cel:/URLs kopiert haben, werden Sie feststellen, dass es gar nicht so kompliziert ist, wie es hier zunächst scheint.

---

gotolonglat

Argumente:

```
time      <Wert>      (Voreinstellung:  1.0)
distance  <Wert>      (Voreinstellung:  5.0)
up        <Vector>    (Voreinstellung:  Y [0 1 0])
longitude <Wert>
latitude  <Wertr>
```

Das `gotolonglat` Kommando bewegt die Ansicht in der angegeben Zeit (in Sekunden) zum aktuell ausgewählten Objekt und stoppt in der angegebenen Entfernung und verwendet die eingestellte Orientierung/Ausrichtung (`up`). Anschließend positioniert es Sie über die angegebenen Koordinaten der Objektoberfläche (Längen- und Breitengrad).

Auch hier ist zunächst die Auswahl eines Objektes mit `select` erforderlich.

Beachten Sie, dass `gotolonglat` nicht das aktuelle Koordinatensystem verändert, sondern das eingestellte Koordinatensystem beibehält.

`time`	Zeitraum in Sekunden, in dem das Objekt angefahren wird.

<table>
<tr><td>distance</td><td>Gibt an, wie weit weg Sie sich von dem Objekt positionieren möchten.
Die Maßeinheit hierfür beträgt: Objektradius plus 1.
Wenn Sie als Wert für distance 6 angeben und der Radius des Objektes 10.000 km beträgt, werden Sie 50.000 km vom Mittelpunkt des Objektes entfernt positioniert. (Anm. d. Übers.: Klingt komisch, is' aber so! ☺)

Am einfachsten berechnen Sie den erforderlichen Wert wie folgt:

Gewünschte Entfernung geteilt durch Objektradius plus 1

Nehmen wir das Beispiel von oben:
Sie möchten 50.000 km von einem Objekt mit einem Radius von 10.000 km entfernt positioniert werden. Dann sieht die Rechnung wie folgt aus:

<code>(50.000 / 10.000) + 1 = </code><code>6</code>

Anderes Beispiel mit der Erde:
Sie möchten 1.000.000 km von der Erde entfernt positioniert werden. Die Erde hat einen Radius von 6.378,1 km. Dann sieht die Rechnung wie folgt aus:

<code>(1000000 / 6,378.1) + 1 = </code><code>157.7865</code>

Vorsicht mit folgenden Werten für distance :

0 (Null) = Mittelpunkt des Objektes. Wenn Sie diesen Wert in der Version 1.3.1 verwenden, kann Celestia anschließend weitere Positionierungen nicht mehr richtig erkennen! Sie sollten deshalb den Wert „Null“ nicht einsetzen.

1 = Oberfläche des Objektes. Wenn Sie die Ansicht ganz genau auf die Oberfläche des Objektes herabsenken (Höhe = 0 km), zeigen einige Grafikkarten nur noch Polygone an. Am besten setzen Sie daher einen Wert etwas oberhalb der Oberfläche ein.</td></tr>
<tr><td>up</td><td>Definiert, welche Achse nach oben zeigt.
X [1 0 0], Y [0 1 0] oder Z [0 0 1]</td></tr>
<tr><td>longitude
latitude</td><td>Diese beiden Argumente (bzw. deren Werte) geben an, über welcher Koordinate der Objektoberfläche Sie sich aufhalten möchten. Die Werte werden in Dezimalzahlen angegeben.

Longitude (Längengrad) ist spezifiziert als ein negativer Wert für die westliche Hemisphäre und als positiver Wert für die östliche Hemisphäre (ein Pluszeichen ist nicht erforderlich).

Latitude (Breitengrad) ist spezifiziert als negativer Wert für die südliche Hemisphäre und als positiver Wert für die nördliche Hemisphäre (ein Pluszeichen ist nicht erforderlich).</td></tr>
</table>

	***Verwenden Sie keinen negativen Wert für das time Argument! Andernfalls hängt sich Celestia auf.***

<table>
<tr>
<td></td>
<td>* Sie können andere Kommandos benutzen, die keine Bewegungen ausführen (z.B. print), während sich die Ansicht zum Ziel begibt.

* Bevor Sie andere Kommandos mit einem time Argument benutzen, müssen Sie erst ein wait Kommando ausführen lassen, welches die gleiche oder eine längere Dauer als das time Argument des gotolonglat Kommandos haben muss. Andernfalls wird das gotolonglat Kommando nicht ausgeführt, stattdessen wird das nächste Kommando mit einem time Argument ausgeführt.</td>
</tr>
</table>

Das nachfolgende Beispiel wählt die Erde aus und positioniert die Ansicht über Seattle, Washington, USA:

```
select      { object "Sol/Earth" }
synchronous { }
gotolonglat { time 5
              distance 3
              up [0 1 0]
              longitude -122
              latitude 47 }

print       { text "Traveling to Seattle, Washington, USA."
              row -3  column 1  duration 5 }
wait        { duration 5 }

print       { text "Hovering over Seattle, Washington, USA."
              row -3  column 1  duration 5 }
wait        { duration 5 }
```

---

labels

Argumente:
```
set   <String>
clear <String>
```

Mit dem Argument set schalten Sie die Bezeichnungen (labels) für die unten genannten Objekte ein und mit dem Argument clear schalten Sie sie wieder ab. Mehrfacheingaben werden in Anführungszeichen gesetzt und mit einem vertikalen Srich (|) getrennt (z.B.: "moons|stars").

- asteroids (Asteroiden)
- comets (Kometen)
- constellations (Sternbilder)
- galaxies (Galaxien)
- moons (Monde)
- planets (Planeten)
- spacecraft (Raumschiffe u. -sonden)
- stars (Sterne)

Die beiden folgenden Beispiele zeigen, wie man Bezeichnungen (`labels`) mit `clear` ausschaltet und mit `set` einschaltet:

```
labels { clear "comets|constellations|galaxies|stars" }
labels {   set "asteroids|moons|planets|spacecraft" }
```

Anm. d. Übers.: In der ersten Zeile werden zunächst die Bezeichnungen für Kometen, Sternbilder, Galaxien und Sterne **ab**geschaltet, in der zweiten Zeile werden die Bezeichnungen für Asteroiden, Monde, Planeten und Raumfahrzeuge **ein**geschaltet.

---

lock

Argumente: keine

Dieses Kommando koppelt zwei Objekte aneinander (siehe auch Abschnitt zum Koordinatensystem).

Das zuerst gewählte Objekt nennt sich **Referenzobjekt**, das zweite (anschließend) ausgewählte nennt sich **Zielobjekt**.
Wenn zwei Objekte als ein Paar miteinander verbunden werden und Sie diese Szenerie umkreisen, bleibt das Zielobjekt am Referenzobjekt angebunden. Die oben links auf dem Monitor angezeigte Entfernung ist der Abstand zwischen diesen beiden Objekten (und nicht wie normalerweise üblich, der Abstand zwischen Ihnen und dem Objekt).

Im Lock-Modus verbleibt Ihre Position auf einer relativen Linie zwischen dem Referenzobjekt und dem Zielobjekt. Wenn das Zielobjekt sich um das Referenzobjekt bewegt, so werden Sie sich ebenso um das Referenzobjekt bewegen.
Wenn das Zielobjekt also z.B. die Sonne ist, bleibt der beleuchtete Teil eines Planeten (seine „Phase“) immer gleich, denn Sie bewegen sich synchron mit der Sonne um diesen Planeten.

Das nachfolgende Beispiel hält Ihre Position mit angemessenem Abstand vom Mittelpunkt der Erde und fixiert sowohl die Sonne als auch die Erde in der Ansicht.

```
select { object "Sol/Earth" }
follow { }
select { object "Sol" }
lock   { }
```

---

lookback

Argumente: keine

Schaltet die Ansicht um 180° um, etwa so, als würden Sie aus dem Rückfenster Ihres Raumschiffes blicken.

Anm. d. Übers.: Wenn Sie das Kommando nochmals anwenden, wird der Blick wieder umgekehrt und Sie schauen wieder aus dem Frontfenster Ihres Raumschiffes.

Beispiel:

```
lookback { }
```

---

mark

Argumente:
```
object <String>
size   <Wert>    (Voreinstellung: 10.0)
color  <Vector>  (Voreinstellung: [1 0 0] - Rot)
symbol <String>  (Voreinstellung: "diamond")
```

Dieses Kommando markiert das entsprechende Objekt mit dem spezifizierten Symbol in der angegeben Größe (`size`) und mit der angegeben Farbe (`color`).
Anm. d. Übers.: Standardmäßig sieht das Symbol so aus: ♢ (im Script als „diamond" bezeichnet) und ist im aktiven Zustand rot (im inaktiven Zustand grün).

`object`	Name des Objektes, das markiert werden soll.
`size`	Durchmesser des Symbols (Angabe in Pixel).

`color`	Definiert die Farbe des Symbols. Der aus drei Werten bestehende Gesamtwert definiert dabei die Anteile von Rot, Grün und Blau (RGB). Zulässige Werte reichen von 0 bis 1, wobei Dezimalwerte mit einer führenden Null (z.B. 0.5) erlaubt sind. Alle Werte über 1 werden schlicht als 1 interpretiert.
	
	Hier einige Beispiele:
	
	`Schwarz:[0 0 0]`
	`Rot:     [1 0 0]` *(Voreinstellung/Standard)*
	`Grün:    [0 1 0]`
	`Blau:    [0 0 1]`
	`Gelb:    [1 1 0]`
	`Weiß:    [1 1 1]`

`symbol`	Definiert das Aussehen des Symbols. Nur folgende String-Werte sind zulässig:
	
	`* diamond` *(Vorgabe/Standard)*
	`* plus`
	`* square`
	`* triangle`
	`* x`

	Wenn das Objekt bereits vorher markiert wurde, müssen Sie diese (bereits vorhandene) Markierung vor Verwendung von `mark` zunächst mit `unmark` aufheben. Auch wenn Sie nicht wissen, ob das Objekt bereits vorher markiert wurde, sollten Sie sicherheitshalber `unmark` anwenden.

Das folgende Beispiel markiert die Erde mit einem grünen „X“:

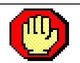
```
unmark { object "Sol/Earth" }
mark   { object "Sol/Earth"
         size 15
         color [0 1 0]
         symbol "x" }
```

move

Argumente:
`duration <Wert>`
`velocity <Vector>`

Bewegt die Kamera (also die Ansicht) mit der angegebenen Geschwindigkeit (`velocity`) in der angegebenen Zeitspanne (`duration`). Ein `wait` Kommando ist für das `move` Kommando nicht erforderlich.

`duration`	Wert in Sekunden.
`velocity`	Geschwindigkeit in Kilometer pro Sekunde (km/s).

	***Verwenden Sie keinen negativen Wert für das `duration` Argument, sonst wird es ignoriert.***

Das Beispiel bewegt die Kamera für 10 Sekunden mit einer Geschwindigkeit von 100.000 km/s:

```
move { duration 10 velocity 100000 }
```

orbit

Argumente:
`duration <Wert>` (Voreinstellung: `1.0`)
`rate     <Wert>` (Voreinstellung: `0.0`)
`axis     <Vector>`

Umläuft das ausgewählte Objekt unter Verwendung des gerade aktuellen Koordinatensystems.

Sie müssen zunächst das `select` Kommando zur Auswahl eines Objektes verwenden und können optional das Kommando `setframe` anwenden, um ein Koordinatensystem zu bestimmen, falls aktuell kein Koordinatensystem definiert ist.

`duration`	Wert in Sekunden.
`rate`	Geschwindigkeit, mit der das Objekt umlaufen wird. Positive und negative Werte bestimmen die Umlaufrichtung (Pluszeichen ist nicht erforderlich).

`axis`	Definiert, um welche Achse der Umlauf vollzogen wird [X, Y, Z]. Der Wert 1 steht für „Ja“, der Wert 0 für „Nein“. Sie können auch mehrere Achsen definieren. Beispiele: Um die X-Achse zu definieren nutzen Sie [1 0 0], für die Y-Achse verwenden Sie [0 1 0] und für die Z-Achse [0 0 1]. Um z.B. die X- **und** Y-Achse zu definieren verwenden Sie [1 1 0].

	***Verwenden Sie keinen negativen Wert für das `duration` Argument, sonst wird es ignoriert.***

Das folgende Beispiel umläuft den Saturn für 12 Sekunden:

```
select { object "Sol/Saturn" }
center { }
goto   { time 3
         distance 8
         up [ 0 1 0 ]
         upframe "equatorial" }
wait   { duration 3 }
orbit  { axis [ 0 1 0 ]
         rate 30
         duration 12 }
```

---

preloadtex

Argumente:
`object <String>`

Lädt die angegebene Textur (Objektoberfläche) vorab von der Festplatte in den Hauptspeicher.

`object`	Der Wert besteht nur aus dem Dateinamen (ohne Angabe des Pfades). Celestia benutzt interne Suchlisten und findet die Datei daher selbstständig. Solche Suchlisten sind oft kontextbezogen. Sie sind also abhängig vom Speicherort der Add-Ons, die die `.ssc`-Dateien enthalten und das Objekt näher definieren. Das bedeutet unter Umständen, dass Celestia nicht die Oberflächentextur lädt, die Sie eigentlich erwartet haben. Beachten Sie außerdem, dass die Dateierweiterung auch ein Platzhalter (*) sein kann, so dass verschiedene Bilddateiformate geladen werden können. Dabei geht Celestia in folgender Reihenfolge vor: `.png`, `.jpg` und `.dds`. Anm. d. Übers.: Findet Celestia also beispielsweise einen zutreffenden Dateinamen mit der Erweiterung .png, wird diese Textur geladen, selbst wenn der gleiche Dateiname auch als .jpg-Datei vorliegen sollte. Sollten Sie also eine Textur mit gleichem Namen und unterschiedlichen Dateierweiterungen haben und möchten, dass z.B. die `.dds`-Datei geladen wird, müssen Sie die Erweiterung angeben (und dürfen keinen Platzhalter verwenden).

Je nach Größe der Datei, sollten Sie dem Kommando `preloadtex` ein `wait` Kommando folgen lassen. Wenn Sie mehrere Texturen oder sehr große Texturen vorladen möchten, sollten Sie auf jeden Fall das `wait` Kommando anfügen. Dabei ist die Dauer der Script-Pause natürlich abhängig von der Größe der Datei(en). Sie müssen also ggf. ausprobieren, wie lange die Script-Pause sein muss.

Das Beispiel lädt die Textur-Datei “mars.jpg” in den Speicher:

```
preloadtex { object "mars.jpg" }
```

Anm. d. Übers.: Wenn Celestia die erst beste Mars-Textur laden soll, würden Sie folgendes eingeben (mit Platzhalter):

```
preloadtex { object "mars.*" }
```

---

print

Argumente:

```
text     <String>
origin   <String>        (Voreinstellung: "bottomleft")
duration <Wert>          (Voreinstellung: 1.0)
row      <Wert>          (Voreinstellung: 0)
column   <Wert>          (Voreinstellung: 0)
```

Dieses Kommando erlaubt Ihnen, auf dem Monitor Text anzuzeigen. Dabei können Sie festlegen, wo der Text erscheinen und wie lange er angezeigt werden soll. Sie **müssen** dem `print` Kommando ein `wait` Kommando folgen lassen, dessen Argument-Wert gleich oder größer als die erforderliche Zeitspanne ist.

Vorsicht: Wenn der Text länger ist, als die Breite des Celestia-Fensters, verschwindet der Text außerhalb des rechten Fensterrandes! Anm. d. Übers.: Sie müssen in einem solchen Fall den Text also schon im Script umbrechen lassen.

`text`	Der anzuzeigende Text muss mit Anführungsstrichen umschlossen werden. Mit der Zeichenfolge "\n" können Sie einen Zeilenumbruch an geeigneter Stellen erzwingen (falls erforderlich).

origin	Gibt die Startposition des ersten Textzeichens an. Nur folgende (nicht in Klammern gesetzte) Werte sind zulässig: * bottom (unten) * bottomleft (unten links) * bottomright (unten rechts) * center (Mitte) * left (Links) * right (Rechts) * top (oben) * topleft (oben links) * topright (oben rechts) Beschreibung der Startpositionen: bottom: unten, eingemittet (jedoch nicht genau mittig zentriert). bottomleft: untere linke Ecke. bottomright: untere rechte Ecke. Sie müssen einen **negativen** Zeilenwert (column) angeben, sonst verschwindet Ihr Text außerhalb der rechten Fensterecke. center: Mittig auf dem Bildschirm (jedoch nicht genau mittig zentriert). left: Linker Rand (auf halber Höhe hin zum unteren Rand). right: Rechter Rand (auf halber Höhe hin zum unteren Rand). Sie müssen einen **negativen** Zeilenwert angeben, sonst verschwindet Ihr Text außerhalb des rechten Fensterrandes. top: oben, eingemittet (jedoch nicht genau mittig zentriert). topleft: Oben links im Bildschirm. topright: Oben rechts im Bildschirm. Sie müssen einen **negativen** Zeilenwert angeben, sonst verschwindet Ihr Text außerhalb des rechten Fensterrandes. Benutzen Sie row und column um den Startpunkt vertikal und horizontal zu positionieren.
duration	Anzeigedauer des Textes (in Sekunden).
row	Definiert die Position der ersten Textzeile (vertikale Position), versetzt von dem im Argument origin angegebenen Startpunkt. Positive Werte setzen die Zeile weiter nach unten (Pluszeichen ist nicht erforderlich) , negative Werte setzen die Zeile weiter nach oben. Beispiel: Wenn Sie den Startpunkt mit left angegeben haben, der Startpunkt aber tatsächlich drei Zeilen höher (als left festlegt) sein soll, würden Sie einen Wert von -3 für row angeben. Soll die Zeile dagegen drei Zeilen tiefer beginnen, würden Sie einen Wert von 3 (ohne Pluszeichen) einsetzen.

`column`	Definiert die Spalten-Position (horizontale Position), an die der erste Buchstabe Ihres Textes beginnen soll. Der Spaltenwert 0 (Null) entspricht dem linken Rand des Bildschirmes.

	Das `print` Kommando muss von einem `wait` Kommando gefolgt werden, dessen Dauer (`duration`) gleich oder länger als die des `print` Kommandos sein muss. Andernfalls wird der Text nicht sichtbar angezeigt.

	***Verwenden Sie keinen negativen Wert für das `duration` Argument, sonst wird es ignoriert.***

Das nachfolgende Beispiel schreibt die Nachricht “Hallo Universum!” drei Zeilen über dem unteren Rand und zwei Spalten vom linken Rand entfernt:

```
print { text "Hallo Universum!"  row -3  column 2 }
wait  { duration 1 }
```

Folgendes Beispiel schreibt das Wort “Hallo” fünf Zeilen über dem unteren Rand des Bildschirmes und beginnt zwei Spalten vom linken Rand entfernt und schreibt in die nächste Zeile das Wort „Universum!“:

```
print { text  "Hallo\nUniversum!"
        origin "left"
        duration 5
        row -5
        column 2 }
wait  { duration 5 }
```

Anm. d. Übers.: Beachten Sie, dass Sie vor und hinter der Zeichenfolge für den erzwungen Zeilenumbruch (`\n`) kein Leerzeichen einfügen brauchen!

---

renderflags

Argumente:

```
set   <String>
clear <String>
```

Mit den Argumenten `set` und `clear` schalten Sie eines oder mehrere der nachfolgenden Objekte und Details ein (`set`) oder ab (`clear`). Dabei können die String-Werte für diese Argumente beliebig kombiniert werden. Mehrere String-Werte werden mit Anführungszeichen umfasst und mit einem senkrechten Strich (|) getrennt. (z.B.: `"orbits|stars"`). In den eckigen Klammern stehen die entsprechenden Tastaturbefehle für den interaktiven Modus außerhalb von Skripten:

- `atmospheres` ...... Atmosphären [Strg+A]
- `automag` .... Auto Magnitude [Strg+Y]
- `boundaries` .... Konstellationsgrenzen [Strg+B]
- `cloudmaps` .... Wolkentexturen [I]

- `comettails .......` Kometenschweife [Strg+T]
- `constellations ...` Sternbilderbezeichnungen [Strg+/]
- `eclipseshadows ...` Finsternisschatten [Strg+E]
- `galaxies .........` Galaxien (anzeigen) [U]
- `grid .............` Erdbasiertes äquatoriales Raster [;]
- `markers ..........` Markierungen [Strg+K]
- `nightmaps ........` Nachtlichtertexturen [Strg+ L]
- `orbits ...........` Orbits der Objekte [O]
- `planets ..........` Planetenbezeichnungen [P]
- `pointstars .......` [nicht mehr in Gebrauch -- siehe `set` Kommando]
- `ringshadows ......` Ringschatten [ohne Tastaturbefehl]
- `stars ............` Sterne [ohne Tastaturbefehl]

Beispiele:

```
renderflags { set "automag|atmospheres|nightmaps" }
renderflags { clear "boundaries|galaxies|markers" }
```

Anm. d. Übers.: Beachten Sie bei der Verwendung des String-Wertes `orbits`, dass diese nur dann über das Script ein- oder ausgeschaltet werden können, wenn Sie im Menü „**Darstellung**“ -> „**Anzeige-Optionen**“ („Render“ -> „View Options“) und dort im Options-Feld „**Orbits/Bezeichnungen**“ („Orbits / Labels“) die Option „**Orbit**“ für das/die entsprechende/n Objekt/e mit einem Häkchen versehen haben! Andernfalls hat das Script keine Auswirkungen bezüglich der Anzeige der Unlaufbahnen für bestimmte Objekte. Das Script schaltet nämlich ausschließlich die grundsätzliche Option „Orbits“ im Options-Feld „**Zeigen**“ („Show“) an oder ab (jedoch nicht, für welche Objekte dies gelten soll).
Da dieser Umstand natürlich sehr unglücklich ist (denn die anderen Benutzer Ihres Scripts müssten gesondert (z.B. mit dem `print` Kommando) auf das Setzen des Häkchens für das betreffende Objekt aufmerksam gemacht werden), bleibt zu hoffen, dass dies in zukünftigen Versionen von Celestia geändert wird (z.B. `object` Argument für das Kommando `renderflags`).

---

rotate

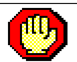
Argumente:
`duration <Wert>` (Voreinstellung: `1.0`)
`rate     <Wert>` (Voreinstellung: `0.0`)
`axis     <Vector>`

Dieses Kommando lässt die Kamera (Ansicht) gemäß dem aktiven Koordinatensystem rotieren. Sie müssen zunächst mit dem `select` Kommando ein Objekt bestimmen und können optional mit dem Kommando `setframe` ein Koordinatensystem festlegen, sofern es nicht bereits eingestellt ist.

`duration`	Zeitdauer (in Sekunden), in der das `rotate` Kommando ausgeführt wird.
`rate`	Geschwindigkeit, mit der das Objekt rotiert. Positive und negative Werte bestimmen die Rotationsrichtung (Pluszeichen ist nicht erforderlich).

`axis`	Definiert, um welche Achse die Rotation erfolgen soll [X, Y, Z]. Der Wert 1 steht für „Ja“, der Wert 0 für „Nein“. Sie können auch mehrere Achsen definieren. Beispiele: Um die X-Achse zu definieren nutzen Sie [1 0 0], für die Y-Achse verwenden Sie [0 1 0] und für die Z-Achse [0 0 1]. Um z.B. die X- **und** Y-Achse zu definieren verwenden Sie [1 1 0].

	Das `rotate` Kommando muss von einem `wait` Kommando gefolgt werden, dessen Dauer (`duration`) gleich oder länger als die des `rotate` Kommandos sein muss.

	***Verwenden Sie keinen negativen Wert für das `duration` Argument, sonst wird es ignoriert.***

Das folgende Beispiel lässt die Kamera (Ansicht) für 5 Sekunden entlang der Z-Achse nach rechts rotieren (positiver `rate`-Wert):

```
rotate { duration 5
         rate 10
         axis [0 0 1] }
wait   { duration 5 }
```

Das gleiche Spielchen, diesmal linksherum (negativer `rate`-Wert):

```
rotate { duration 5
         rate -10
         axis [0 0 1] }
wait   { duration 5 }
```

select

Argument:
`object <String>`

Wählt das Objekt aus, um andere Kommandos darauf anwenden zu können.

Wenn Sie sich *außerhalb* unseres Sonnensystems befinden und ein Objekt *innerhalb* unseres Sonnensystem auswählen lassen möchten, müssen Sie dem Objektnamen den Teil-String "`Sol/`" voranstellen, wie es weiter unten im Beispiel dargestellt ist.

`object`	Name des Objektes, das ausgewählt werden soll.

	Wenn das Objekt nicht existiert (oder Sie den Namen falsch geschrieben haben), wird das Script einfach mit dem nächsten Kommando fortfahren, ohne ein Objekt auszuwählen. Auch ein zuvor bereits ausgewählte Objekt ist dann nicht mehr aktiv.

Beispiele:

```
select { object "M33" }      #(Galaxie M33)
select { object "Mars" }     #(innerhalb des Sonnensystems)
select { object "Sol/Mars" } #(außerhalb des Sonnensystems)
```

---

set

Argumente:
`name  <String>`
`value <Wert>` oder `<String>` (Voreinstellung: `0.0` oder `""`)

Setzen Sie einen der gelisteten Einstellungspunkte als Wert für das `value` Argument für den `name`–Parameter ein:

`name`	Folgende String-Werte sind zulässig:
	
	`* MinOrbitSize`
	`* AmbientLightLevel`
	`* FOV`
	`* StarDistanceLimit`
	`* StarStyle`
	
	Hinter diesen Strings verbirgt sich folgendes:
	
	`MinOrbitSize`: Dieser String wird benutzt, wenn Celestia die Umlaufbahnen anzeigen soll und diese Funktion eingeschaltet ist. Grundsätzlich werden die Umlaufbahnen nicht angezeigt, wenn das entspr. Objekt sehr weit weg ist. `MinOrbitSize` bestimmt nun den Mindestradius (in Pixel), den die Umlaufbahnen haben müssen, damit Celestia diese darstellt.
	
	`AmbientLightLevel`: Hiermit kann eine Helligkeitsstufe des Streulichtes festgelegt werden. Dabei reichen die Werte von 0.0 (wenig) bis 1.0 (höchste Stufe).
	
	`FOV`: Hiermit kann das Blickfeld eingestellt werden. Celestia berechnet dieses Blickfeld auch automatisch und richtet sich dabei nach der Größe des Bildschirmes bzw. des Fensters (in dem Celestia läuft).
	
	`StarDistanceLimit`: Legt die weiteste Entfernung fest, in der Celestia noch Sterne darstellt. Die Voreinstellung (Standard) ist 1.000.000 (Lichtjahre).
	
	`StarStyle`: Stellt ein, mit welchem Aussehen die Sterne dargestellt werden sollen. Nur die folgenden String-Werte sind zulässig:
	
	`* fuzzypoints`
	`* points`
	`* scaleddiscs`

`value`	Gibt den (String-) Wert an, der dem Parameter `name` zugeordnet werden soll.

Beispiele:

```
set { name "FOV"       value 35.5 }
set { name "StarStyle" value "points" }
```

setfaintestautomag45deg

Argument:
magnitude <Wert> (Voreinstellung: 8.5)

Stellt die niedrigste Magnitude (scheinbare Helligkeit) von Sternen ein, wenn der Modus „Auto-Magnitude“ (mit einem limitierten Blickfeld von 45°) eingeschaltet ist.

magnitude	Setzt die Magnitude auf den Wert, bei dem Sterne sichtbar sein sollen. Erlaubt sind Werte von 1.0 bis 15.0. Anm. d. Übers.: Höhere Werte (also z.B. 20.0) können eingesetzt werden, ohne dass ein Scriptfehler verursacht oder das Kommando übergangen wird. Allerdings haben höhere Werte keine Auswirkungen; Celestia stellt dann einfach den Maximalwert von 15.0 ein.

Beispiel:

```
setfaintestautomag45deg { magnitude 9.0 }
```

setframe

Argumente:
ref <String>
target <String>
coordsys <String> (Voreinstellung: "universal")

Dieses Kommando stellt das aktive Koordinatensystem ein. Lesen Sie hierzu näheres im Abschnitt zu den in Celestia verwendbaren Koordinatensystemen.

ref	Definiert das **Referenz**objekt (also das zuerst gewählte Objekt).
target	Optional – Definiert das **Ziel**objekt (also das anschließend ausgewählte Objekt) für ein Koordinatensystem mit zwei Objekten (wie z.B. lock).
coordsys	Legt das anzuwendende Koordinatensystem fest. Nur folgende Werte (Koordinatensystem) sind zulässig: * chase * ecliptical * equatorial * geographic * lock * observer * universal

Es kann immer nur ein Koordinatensystem zur gleichen Zeit aktiv sein:

- `chase` (Chase)
- `equatorial`
- `ecliptical` (Follow)
- `geographic` (Sync Orbit)
- `lock` (Lock)
- `observer`
- `universal` (ohne Bindungen)

Das nachfolgende Beispiel ruft das Koordinatensystem `lock` auf, welches die Erde und den Mond miteinander auf dem Monitor koppelt:

```
setframe { ref "Sol/Earth"
           target "Sol/Earth/Moon"
           coordsys "lock" }
```

---

setorientation

Argumente:

```
angle <Wert>
axis  <Vector>
```

-oder-

```
ow <Wert>
ox <Wert>
oy <Wert>
oz <Wert>
```

Stellt die Kamera-Ausrichtung ein.
Wenn Sie eine Position duplizieren wollen, die auf einem Lesezeichen (Bookmark) oder einer Cel://URL basiert, müssen Sie außerdem das korrekte Koordinatensystem und andere Parameter einstellen (siehe auch `seturl`).

`angle`	Diesen Wert können Sie aus einem Lesezeichen entnehmen (falls Sie solche angelegt haben), die in der Datei `fovorites.cel` gespeichert sind.
`axis`	Diesen Wert können Sie ebenfalls aus einem Lesezeichen entnehmen (falls Sie solche angelegt haben), die in der Datei `fovorites.cel` gespeichert sind.

Die Werte `ow`, `ox`, `oy` und `oz` repräsentieren die Winkel und Achsen, die in Cel://URLs gespeichert sind. Sie können aus den folgenden Cel://URL-Werten entnommen werden:

`&ow=` `&ox=` `&oy=` und `&oz=`

Beispiele:

```
setorientation {
   angle 0.945208
   axis  [ 0.81466 -0.570975 -0.101573 ] }

setorientation {
   ow 0.090610
   ox -0.494683
   oy 0.860207
   oz -0.084397 }
```

---

setposition

Argumente:
`base   <Vector>`
`offset <Vector>`
-oder-
`x <Base64>`
`y <Base64>`
`z <Base64>`

Befördert die Kamera (Ansicht) in eine bestimmte Position im dreidimensionalen Universum.
Wenn Sie eine Position duplizieren wollen, die auf einem Lesezeichen (Bookmark) oder einer Cel://URL basiert, müssen Sie außerdem das korrekte Koordinatensystem und andere Parameter einstellen (siehe auch `seturl`).

`base`	Diesen Wert können Sie aus einem Lesezeichen entnehmen (falls Sie solche angelegt haben), die in der Datei `fovorites.cel` gespeichert sind.
`offset`	Diesen Wert können Sie ebenfalls aus einem Lesezeichen entnehmen (falls Sie solche angelegt haben), die in der Datei `fovorites.cel` gespeichert sind.

Die Werte `x`, `y` und `z` repräsentieren die `base` und `offset` Werte, die in Cel://URLs gespeichert sind. Sie können aus den folgenden Cel://URL-Werten entnommen werden:
`&x=` `&y=` und `&z=`
Beide Beispiele positionieren die Kamera an die exakt gleiche Position im Universum außerhalb der Milchstraße:

```
setposition {
    base [ -64132.43  47355.11  196091.57 ]
  offset [ 0  0  -1.52e-005 ] }

setposition {
  x "AMDCXoJK/3+IyGgR8f///w"
  y "BvP2LRdAAAD/n5UGCw"
  z "VUrGoeQJ+/8DyvenLQ" }
```

setsurface

Argument:
`name <String>`

Dieses Kommando erlaubt Ihnen die Auswahl einer alternativen Oberflächentextur für das gewählte Objekt. Allerdings muss diese alternative Oberflächentextur (alternate surface texture) in der Datei `solarsys.ssc` für das entspr. Objekt definiert sein (oder von Ihnen vor der Verwendung von `setsurface` definiert werden).

Wenn Sie also beispielsweise eine alternative Oberfläche für die Erde mit der Bezeichnung "Earth-2" schaffen möchten und der entsprechende Dateiname für diese alternative Oberfläche heißt `earth2.jpg`, würden Sie folgenden Eintrag in der Datei `solarsys.ssc` nach der schließenden Klammer hinter dem Eintrag „Earth" „Sol" vornehmen müssen:

```
AltSurface "Earth-2" "Sol/Earth"
{
  Texture "earth2.jpg"
}
```

In diesem Beispiel ist "Earth-2" die Bezeichnung für die alternative Oberfläche, die Sie im Argument `name` für das `setsurface` Kommando eintragen müssten:

```
setsurface { name "Earth-2" }
```

`name`	Definiert den Namen (die Bezeichnung) der alternativen Oberfläche, die in der Datei `solarsys.ssc` vorgegeben ist. **Achtung:** Nicht den Datei-Namen der alternativen Oberfläche eintragen!

	Wie so oft auch bei anderen Kommandos, müssen Sie zunächst mit `select` ein Objekt auswählen, bevor Sie das `setsurface` Kommando einsetzen können.

Um die in Celestia enthaltenen "limit of knowledge"-Texturen (näheres hierzu lesen Sie in meinem deutschsprachigen Benutzerhandbuch nach) statt der interpretierenden Oberflächen einzusetzen, nutzen Sie das `setsurface` Kommando wie folgt:

```
setsurface { name "limit of knowledge" }
```

setvisibilitylimit

Argument:
`magnitude <Wert>` (Voreinstellung: `6.0`)

Stellt die Magnitude (scheinbare Helligkeit) der anzuzeigenden Sterne ein, wenn der Auto-Magnitude-Modus (AutoMag) **aus**geschaltet (also nicht aktiviert) ist. Die Abstufungen reichen von 1.0 bis 15.0.

magnitude	Definiert die Magnitude, ab der Sterne sichtbar werden, wenn AutoMag **nicht** aktiviert ist.

Beispiel:

```
setvisibilitylimit { magnitude 6.5 }
```

---

seturl

Argument:
`url <String>`

Bewegt die Ansicht (Kamera) zu dem Ort, der in einer Cel://URL abgespeichert ist, die Sie mit [Strg+C] in die Zwischenablage kopieren können.

url	Definiert die Cel://URL, die benutzt werden soll.

	Wenn Sie das `seturl` Kommando einsetzen, werden **alle** in der Cel://URL gesicherten Einstellungen in das laufende Programm übertragen.

Beispiel:

```
seturl { url
 "cel://Follow/Sol/2002-09-
01T16:58:51.32378?x=AAAAAMCrktOdILb5/////w&y=CkxgAADEXts7E7HX/////w&z
=VcpQAAAAAAAAAMIy/////w&ow=0.090610&ox=-0.494683&oy=0.860207&oz=-
0.084397&select=Sol&fov=49.499992&ts=1.000000&ltd=0&rf=6043&lm=0" }
```

Nachfolgend einige Beispiele, die aus Cel://URLs „generiert“ wurden:
(Kopieren Sie jeweils den vollständige String in die Adresszeile Ihres Browsers und drücken die Eingabetaste. Daraufhin wird Celestia gestartet und die Szene angezeigt)

Sonnenfinsternis:

```
cel://Freeflight/2004-01-
23T14:16:37.53441?x=QGcgesLkFj+1DA&y=IGXjwESpGv//////////w&z=Fbs1esf
Xhe7y/////////w&ow=0.231217&ox=-
0.566846&oy=0.772273&oz=0.169759&select=Sol:Earth:Moon&fov=4.098599&t
s=0.000000&ltd=0&rf=68503&lm=49152
```

Mond und Erde kreuzen die Sonne:

```
cel://PhaseLock/Sol:Earth/Sol:Earth:Moon/2004-01-
21T17:41:47.40987?x=AACt9VF45j26Cg&y=9W04W6+z8kE&z=VUqNIERs1NWX/P////
///w&ow=0.153437&ox=0.784752&oy=0.560930&oz=-
0.214428&select=Sol:Earth:Moon&fov=0.000976&ts=100.000000&ltd=0&rf=29
67&lm=49158
```

ISS rotiert vor der Sonne:

```
cel://PhaseLock/Sol:Earth:ISS/Sol/2004-01-
21T20:32:57.17702?x=+BwvJlcrcqO1DA&y=YMaF/EDm6///////////w&z=EuvkSMO
MmK7y/////////w&ow=0.221298&ox=0.542102&oy=0.796562&oz=-
0.150474&select=Sol:Earth:ISS&fov=0.479170&ts=10.000000&ltd=0&rf=6850
3&lm=49158
```

Sonnenaufgang, ISS über der Erde:

```
cel://PhaseLock/Sol:Earth:ISS/Sol/2004-01-
21T23:37:16.65668?x=VXeuKd/4q5u1DA&y=4oqqiviL7P//////////w&z=mNTz/Tx
EPLPy/////////w&ow=0.129734&ox=-
0.846240&oy=0.441594&oz=0.268405&select=Sol&fov=8.524273&ts=10.000000
&ltd=0&rf=2967&lm=49152
```

Sonnenaufgang aus Sicht der ISS:

```
cel://PhaseLock/Sol:Earth:ISS/Sol/2004-01-
21T23:37:44.04313?x=m6kFYyBup5u1DA&y=6WrIUIH+6///////////w&z=UIemtce
AQLPy/////////w&ow=0.123742&ox=-
0.854558&oy=0.441435&oz=0.244036&select=Sol&fov=1.972328&ts=1.000000&
ltd=0&rf=2967&lm=49152
```

Sonnenaufgang aus Sicht der ISS (2):

```
cel://PhaseLock/Sol:Earth:ISS/Sol/2004-01-
21T23:37:43.95277?x=Fi7Q079vp5u1DA&y=NGnFKfP+6///////////w&z=Prk+BbF
5QLPy/////////w&ow=0.123742&ox=-
0.854558&oy=0.441435&oz=0.244036&select=Sol&fov=1.972328&ts=1.000000&
ltd=0&rf=2967&lm=49152
```

Sonnenuntergang, ISS über der Erde:

```
cel://PhaseLock/Sol:Earth:ISS/Sol/2004-01-
22T00:33:37.32971?x=dyxw16Pmkpm1DA&y=oLa0vUflEQ&z=0bGSliRpg7Ty//////
//w&ow=0.129839&ox=-
0.846187&oy=0.441496&oz=0.268683&select=Sol&fov=8.524272&ts=10.000000
&ltd=0&rf=2967&lm=49152
```

---

synchronous

Argumente: keine

Umläuft das ausgewählte Objekt im synchronen Orbit-Modus. Dieser Modus aktiviert das Koordinatensystem `geographic` (siehe dazu Abschnitt Koordinatensystem).
Das geographische Koordinatensystem erlaubt Ihnen, in einer stationären (geosynchronen) Umlaufbahn über dem ausgewählten Objekt zu verweilen (dies gilt nicht nur für die Erde, sondern auch für andere Objekte). Während das Objekt unter Ihnen rotiert, folgen Sie dieser Rotation. Damit bleiben Sie scheinbar an der selben Stelle über dem Objekt. Im interaktiven Modus nennt sich dies „SyncOrbit“, Sie erreichen diesen Modus mit per Tastaturbefehl **[Y]** oder über das Menü „Navigation“.

Sie müssen mit select zunächst ein Objekt auswählen, bevor Sie das Kommando synchronous anwenden.



Es kann immer nur ein Koordinatensystem zur gleichen Zeit aktiv sein:

- chase (Chase)
- equatorial
- ecliptical (Follow)
- geographic (Sync Orbit)
- lock (Lock)
- observer
- universal (ohne Bindungen)

Beispiel:

```
select { object "Sol/Earth" }
synchronous { }
```

---

time

Argumente:

jd <Wert> (Julianisches Datum)

-oder-

utc <String> (Weltzeit, UTC=Universal Time)

Stellt das Datum und die Uhrzeit als Julianisches Datum (eine Dezimalzahl) oder als Weltzeit im Format YYYY-MM-DDTHH:MM:SS.SSSS dar.

Anm. d. Übers.: Das Datum 01.04.2004, 15:30 Uhr würde also wie folgt aussehen:

2004-04-01T15:30:00.00000.

jd	Eingabe eines gültigen Julianischen Datums erforderlich.
utc	Eingabe der Weltzeit erforderlich. Format: YYYY-MM-DDTHH:MM:SS.SSSSS

Um ein “normales” Datum in das julianische Format umzurechnen, können Sie den Umrechner der U.S. Navy verwenden, den Sie unter folgender URL erreichen:

http://aa.usno.navy.mil/data/docs/JulianDate.html.

Beispiele, wie Sie das Datum im Script auf den 11.08.2003 um 9:29:24 UTC einrichten:

```
time { jd 2452862.89542 }
 -oder-
time { utc "2003-08-11T09:29:24.0000" }
```

Anm. d. Übers. : Beachten Sie, dass Sie den String-Wert für das Argument utc in Anführungsstriche setzen müssen!

timerate

Argument:
`rate <Wert>` (Voreinstellung: `1.0`)

Mit diesem Kommando stellen Sie den Multiplikator für die Geschwindigkeit des Zeitverlaufes ein.

`rate`	Definiert den Zeitmultiplikator (z.B. x100 = 100mal schneller oder langsamer). Besondere Werte sind: 0 = Zeit anhalten 1 = Auf Realzeitablauf zurücksetzen Negative Werte lassen die Zeit rückwärts laufen.

Das Beispiel lässt die Zeit 1.000 Mal schneller ablaufen:

```
timerate { rate 1000 }
```

track

Argumente: keine

Mit diesem Kommando folgen Sie dem ausgewählten Objekt nach und halten es in der Mitte des Monitors zentriert.

Auch hier muss mit dem `select` Kommando vorher ein Objekt ausgewählt werden.

Das nachfolgende Beispiel hebt zunächst evtl. vorhandene Bindungen und die Selektion von Objekten auf, wählt dann die Erde als Objekt aus und folgt dieser dann nach.
Die Erde wird von Ihrer Ansicht aus mit der tatsächlichen Geschwindigkeit, die sie im Weltraum hat, davonrasen, aber das `track` Kommando wird die Erde in der Mitte Ihres Monitores halten. Das Scriptbeispiel demonstriert dies mit einem 1.000fach erhöhten Zeitablauf:

```
cancel   { }
select   { object "Sol/Earth" }
goto     { time 3 distance 7 }
wait     { duration 5 }
track    { }
timerate { rate 1000 }
```

Wenn Sie möchten, dass der Abstand zwischen Ihnen und der Erde unverändert bleibt, fügen Sie **hinter** dem `track` Kommando das Kommando `follow` an.

Wenn für das ausgewählte Objekt das `track` Kommando aktiviert ist, können Sie ab Celestia 1.3.1 den `track`-Modus mit dem folgenden Code aufheben:

```
select { object "" }
track  { }
```

	Das Kommando `cancel` stellt das Koordinatensystem auf `universal` zurück. Wenn Sie also ein bestimmtes Koordinatensystem anwenden möchten, müssen Sie dies mit dem Kommando `setframe` aufrufen, bevor Sie `track` verwenden.

---

unmark

Argument:
`object <String>`

Wenn ein Objekt vorher markiert wurde, können Sie die Markierung für dieses Objekt mit diesem Kommando aufheben.

`object`	Definiert das Objekt, für das Sie die Markierung aufheben möchten.

Beispiel:

```
unmark { object "Sol/Earth" }
```

---

unmarkall

Argument: keine

Dieses Kommando entfernt alle vorher angebrachten Markierungen von **allen** Objekten und schaltet die Anzeige von Markierungen aus.

Beispiel:

```
unmarkall { }
```

---

wait

Argument:
`duration <Wert>` (Voreinstellung: `1.0`)

Hält das Script für die Dauer der im `duration` Argument spezifizierten Zeit (in Sekunden) an.

	***Verwenden Sie keinen negativen Wert für das `duration` Argument, sonst wird es ignoriert.***

Das folgende Beispiel hält das Script für 15 ¼ Sekunden an:

```
wait { duration 15.25 }
```

Veraltetes Celestia Script Kommando

Das nachfolgende Kommando wird inzwischen in Celestia nicht mehr verwendet, weil es durch ein neueres (und verbessertes) ersetzt wurde:

setambientlight

`brightness <Wert>` (Voreinstellung: `0.0`)

Stellt die Helligkeit des Streulichts ein.

Dieses Kommando wurde ersetzt durch die Wertangabe im Parameter `name` im Kommando `set`.